

libstdc++

Generated by Doxygen 1.7.0

Tue Feb 1 2011 16:16:44

Contents

1	Todo List	2
2	Module Documentation	9
2.1	Extensions	9
2.1.1	Detailed Description	10
2.2	SGI	10
2.2.1	Detailed Description	13
2.2.2	Function Documentation	14
2.3	Containers	24
2.3.1	Detailed Description	24
2.4	Sequences	25
2.4.1	Detailed Description	26
2.5	Associative	27
2.5.1	Detailed Description	27
2.6	Unordered Associative	28
2.6.1	Detailed Description	28
2.7	Diagnostics	29
2.7.1	Detailed Description	29
2.8	Concurrency	30
2.8.1	Detailed Description	30
2.9	Exceptions	31
2.9.1	Detailed Description	33
2.9.2	Typedef Documentation	33
2.9.3	Function Documentation	34
2.10	Time	37
2.10.1	Detailed Description	37
2.11	Complex Numbers	37
2.11.1	Detailed Description	42
2.11.2	Function Documentation	42
2.12	Condition Variables	55

2.12.1 Detailed Description	55
2.12.2 Enumeration Type Documentation	55
2.13 Futures	56
2.13.1 Detailed Description	58
2.13.2 Enumeration Type Documentation	58
2.13.3 Function Documentation	59
2.14 I/O	60
2.14.1 Detailed Description	63
2.14.2 Typedef Documentation	63
2.15 Memory	68
2.15.1 Detailed Description	68
2.16 Pointer Abstractions	68
2.16.1 Detailed Description	71
2.16.2 Function Documentation	71
2.17 Mutexes	73
2.17.1 Detailed Description	74
2.17.2 Function Documentation	74
2.18 Numerics	76
2.18.1 Detailed Description	77
2.19 Rational Arithmetic	77
2.19.1 Detailed Description	78
2.20 Threads	79
2.20.1 Detailed Description	79
2.21 Utilities	80
2.21.1 Detailed Description	80
2.22 Numeric Arrays	81
2.22.1 Detailed Description	91
2.22.2 Function Documentation	91
2.23 Mathematical Special Functions	116
2.23.1 Detailed Description	119
2.23.2 Function Documentation	120

2.24	Decimal Floating-Point Arithmetic	125
2.24.1	Detailed Description	125
2.25	Binder Classes	125
2.25.1	Detailed Description	126
2.25.2	Function Documentation	127
2.26	Algorithms	128
2.26.1	Detailed Description	128
2.27	Mutating	129
2.27.1	Function Documentation	132
2.28	Non-Mutating	153
2.28.1	Function Documentation	156
2.29	Sorting	170
2.29.1	Function Documentation	173
2.30	Set Operation	193
2.30.1	Detailed Description	194
2.30.2	Function Documentation	194
2.31	Binary Search	201
2.31.1	Detailed Description	202
2.31.2	Function Documentation	202
2.32	Allocators	207
2.32.1	Detailed Description	208
2.33	Atomics	208
2.33.1	Detailed Description	217
2.33.2	Define Documentation	217
2.33.3	Typedef Documentation	218
2.33.4	Enumeration Type Documentation	224
2.33.5	Function Documentation	224
2.34	Hashes	225
2.34.1	Detailed Description	225
2.35	Locales	225
2.35.1	Detailed Description	228

2.36	Random Number Generation	228
2.36.1	Detailed Description	229
2.36.2	Function Documentation	229
2.37	Regular Expressions	229
2.37.1	Detailed Description	235
2.37.2	Typedef Documentation	235
2.37.3	Function Documentation	237
2.38	Function Objects	268
2.38.1	Detailed Description	269
2.38.2	Function Documentation	270
2.39	Arithmetic Classes	270
2.39.1	Detailed Description	271
2.40	Comparison Classes	271
2.40.1	Detailed Description	272
2.41	Boolean Operations Classes	272
2.41.1	Detailed Description	272
2.42	Negators	273
2.42.1	Detailed Description	273
2.42.2	Function Documentation	274
2.43	Adaptors for pointers to functions	274
2.43.1	Detailed Description	275
2.43.2	Function Documentation	275
2.44	Adaptors for pointers to members	276
2.44.1	Detailed Description	277
2.45	Heap	277
2.45.1	Function Documentation	279
2.46	Iterators	284
2.46.1	Detailed Description	288
2.46.2	Function Documentation	288
2.47	Iterator Tags	291
2.47.1	Detailed Description	292

2.48	Strings	292
2.48.1	Typedef Documentation	293
2.49	Policy-Based Data Structures	293
2.49.1	Detailed Description	294
2.50	Metaprogramming	294
2.50.1	Define Documentation	300
2.50.2	Typedef Documentation	300
2.51	Random Number Generators	301
2.51.1	Detailed Description	303
2.51.2	Typedef Documentation	303
2.51.3	Function Documentation	304
2.52	Random Number Distributions	308
2.53	Uniform Distributions	308
2.53.1	Function Documentation	309
2.54	Normal Distributions	312
2.54.1	Function Documentation	314
2.55	Bernoulli Distributions	317
2.55.1	Function Documentation	318
2.56	Poisson Distributions	322
2.56.1	Function Documentation	324
2.57	Random Number Utilities	330
3	Directory Documentation	332
3.1	include/backward/ Directory Reference	332
3.2	include/x86_64-unknown-linux-gnu/bits/ Directory Reference	334
3.3	include/bits/ Directory Reference	336
3.4	include/debug/ Directory Reference	340
3.5	include/decimal/ Directory Reference	341
3.6	include/ext/pb_ds/detail/ Directory Reference	342
3.7	/mnt/share/src/gcc.git-trunk/libstdc++-v3/doc/ Directory Reference	343
3.8	/mnt/share/src/gcc.git-trunk/libstdc++-v3/doc/doxygen/ Directory Reference	344

3.9	include/ext/ Directory Reference	345
3.10	/mnt/share/src/gcc.git-trunk/ Directory Reference	347
3.11	include/profile/impl/ Directory Reference	348
3.12	include/ Directory Reference	351
3.13	/mnt/share/src/gcc.git-trunk/libstdc++-v3/ Directory Reference	354
3.14	/mnt/share/src/gcc.git-trunk/libstdc++-v3/libsupc++/ Directory Reference	355
3.15	include/parallel/ Directory Reference	356
3.16	include/ext/pb_ds/ Directory Reference	359
3.17	include/profile/ Directory Reference	361
3.18	/mnt/share/src/ Directory Reference	362
3.19	include/tr1/ Directory Reference	363
3.20	include/x86_64-unknown-linux-gnu/ Directory Reference	364
4	Namespace Documentation	364
4.1	__gnu_cxx Namespace Reference	364
4.1.1	Detailed Description	387
4.1.2	Function Documentation	387
4.2	__gnu_cxx::__detail Namespace Reference	400
4.2.1	Detailed Description	401
4.2.2	Function Documentation	401
4.3	__gnu_cxx::typelist Namespace Reference	402
4.3.1	Detailed Description	402
4.3.2	Function Documentation	402
4.4	__gnu_debug Namespace Reference	403
4.4.1	Detailed Description	409
4.4.2	Function Documentation	409
4.5	__gnu_internal Namespace Reference	413
4.5.1	Detailed Description	413
4.6	__gnu_parallel Namespace Reference	413
4.6.1	Detailed Description	429
4.6.2	Typedef Documentation	429

4.6.3	Enumeration Type Documentation	430
4.6.4	Function Documentation	432
4.6.5	Variable Documentation	480
4.7	__gnu_pbds Namespace Reference	480
4.7.1	Detailed Description	483
4.8	__gnu_profile Namespace Reference	483
4.8.1	Detailed Description	488
4.8.2	Typedef Documentation	488
4.8.3	Function Documentation	489
4.9	__gnu_sequential Namespace Reference	489
4.9.1	Detailed Description	490
4.10	abi Namespace Reference	490
4.10.1	Detailed Description	490
4.11	std Namespace Reference	490
4.11.1	Detailed Description	639
4.11.2	Typedef Documentation	639
4.11.3	Enumeration Type Documentation	640
4.11.4	Function Documentation	641
4.11.5	Variable Documentation	739
4.12	std::__debug Namespace Reference	740
4.12.1	Detailed Description	747
4.12.2	Function Documentation	747
4.13	std::__detail Namespace Reference	748
4.13.1	Detailed Description	749
4.14	std::__parallel Namespace Reference	749
4.14.1	Detailed Description	774
4.15	std::__profile Namespace Reference	774
4.15.1	Detailed Description	781
4.15.2	Function Documentation	781
4.16	std::chrono Namespace Reference	782
4.16.1	Detailed Description	785

4.16.2	Typedef Documentation	785
4.16.3	Function Documentation	786
4.17	std::decimal Namespace Reference	787
4.17.1	Detailed Description	798
4.17.2	Function Documentation	798
4.18	std::placeholders Namespace Reference	798
4.18.1	Detailed Description	799
4.19	std::regex_constants Namespace Reference	799
4.19.1	Detailed Description	801
4.19.2	Typedef Documentation	801
4.19.3	Enumeration Type Documentation	801
4.19.4	Function Documentation	802
4.19.5	Variable Documentation	804
4.20	std::rel_ops Namespace Reference	809
4.20.1	Detailed Description	809
4.20.2	Function Documentation	809
4.21	std::this_thread Namespace Reference	811
4.21.1	Detailed Description	811
4.21.2	Function Documentation	811
4.22	std::tr1 Namespace Reference	812
4.22.1	Detailed Description	816
4.23	std::tr1::__detail Namespace Reference	816
4.23.1	Detailed Description	816
5	Class Documentation	816
5.1	__cxxabiv1::__forced_unwind Class Reference	816
5.1.1	Detailed Description	817
5.2	__gnu_cxx::__common_pool_policy< _PoolTp, _Thread > Struct Template Reference	817
5.2.1	Detailed Description	817
5.3	__gnu_cxx::__detail::__mini_vector< _Tp > Class Template Reference	817
5.3.1	Detailed Description	818

5.4	__gnu_cxx::__detail::_Bitmap_counter< _Tp > Class Template Reference	818
5.4.1	Detailed Description	819
5.5	__gnu_cxx::__detail::_Ffit_finder< _Tp > Class Template Reference	820
5.5.1	Detailed Description	821
5.5.2	Member Typedef Documentation	821
5.6	__gnu_cxx::__mt_alloc< _Tp, _Poolp > Class Template Reference	821
5.6.1	Detailed Description	823
5.7	__gnu_cxx::__mt_alloc_base< _Tp > Class Template Reference	823
5.7.1	Detailed Description	824
5.8	__gnu_cxx::__per_type_pool_policy< _Tp, _PoolTp, _Thread > Struct Template Reference	824
5.8.1	Detailed Description	824
5.9	__gnu_cxx::__pool< false > Class Template Reference	825
5.9.1	Detailed Description	826
5.10	__gnu_cxx::__pool< true > Class Template Reference	826
5.10.1	Detailed Description	827
5.11	__gnu_cxx::__pool_alloc< _Tp > Class Template Reference	827
5.11.1	Detailed Description	829
5.12	__gnu_cxx::__pool_alloc_base Class Reference	829
5.12.1	Detailed Description	830
5.13	__gnu_cxx::__pool_base Struct Reference	831
5.13.1	Detailed Description	832
5.14	__gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc > Class Template Reference	832
5.14.1	Detailed Description	834
5.15	__gnu_cxx::__scoped_lock Class Reference	835
5.15.1	Detailed Description	835
5.16	__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > Class Template Reference	835
5.16.1	Detailed Description	840
5.16.2	Constructor & Destructor Documentation	840

5.16.3	Member Function Documentation	845
5.16.4	Member Data Documentation	903
5.17	<code>__gnu_cxx::_Caster<_ToType></code> Struct Template Reference	904
5.17.1	Detailed Description	904
5.18	<code>__gnu_cxx::_Char_types<_CharT></code> Struct Template Reference	904
5.18.1	Detailed Description	905
5.19	<code>__gnu_cxx::_ExtPtr_allocator<_Tp></code> Class Template Reference	905
5.19.1	Detailed Description	906
5.20	<code>__gnu_cxx::_Invalid_type</code> Struct Reference	907
5.20.1	Detailed Description	907
5.21	<code>__gnu_cxx::_Pointer_adapter<_Storage_policy></code> Class Template Reference	907
5.21.1	Detailed Description	909
5.22	<code>__gnu_cxx::_Relative_pointer_impl<_Tp></code> Class Template Reference	910
5.22.1	Detailed Description	910
5.23	<code>__gnu_cxx::_Relative_pointer_impl<const _Tp></code> Class Template Reference	910
5.23.1	Detailed Description	911
5.24	<code>__gnu_cxx::_Std_pointer_impl<_Tp></code> Class Template Reference	911
5.24.1	Detailed Description	911
5.25	<code>__gnu_cxx::_Unqualified_type<_Tp></code> Struct Template Reference	912
5.25.1	Detailed Description	912
5.26	<code>__gnu_cxx::annotate_base</code> Struct Reference	912
5.26.1	Detailed Description	913
5.27	<code>__gnu_cxx::array_allocator<_Tp, _Array></code> Class Template Reference	913
5.27.1	Detailed Description	915
5.28	<code>__gnu_cxx::array_allocator_base<_Tp></code> Class Template Reference	915
5.28.1	Detailed Description	916
5.29	<code>__gnu_cxx::binary_compose<_Operation1, _Operation2, _-</code> <code>Operation3></code> Class Template Reference	916
5.29.1	Detailed Description	917
5.29.2	Member Typedef Documentation	917

5.30	<code>__gnu_cxx::bitmap_allocator< _Tp ></code> Class Template Reference . . .	918
5.30.1	Detailed Description	919
5.30.2	Member Function Documentation	919
5.31	<code>__gnu_cxx::char_traits< _CharT ></code> Struct Template Reference	920
5.31.1	Detailed Description	922
5.32	<code>__gnu_cxx::character< V, I, S ></code> Struct Template Reference	922
5.32.1	Detailed Description	923
5.33	<code>__gnu_cxx::condition_base</code> Struct Reference	923
5.33.1	Detailed Description	923
5.34	<code>__gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 ></code> Struct Template Reference	924
5.34.1	Detailed Description	924
5.35	<code>__gnu_cxx::constant_unary_fun< _Result, _Argument ></code> Struct Template Reference	924
5.35.1	Detailed Description	925
5.36	<code>__gnu_cxx::constant_void_fun< _Result ></code> Struct Template Reference	925
5.36.1	Detailed Description	926
5.37	<code>__gnu_cxx::debug_allocator< _Alloc ></code> Class Template Reference . .	926
5.37.1	Detailed Description	927
5.38	<code>__gnu_cxx::enc_filebuf< _CharT ></code> Class Template Reference	927
5.38.1	Detailed Description	931
5.38.2	Member Typedef Documentation	931
5.38.3	Member Function Documentation	932
5.38.4	Member Data Documentation	950
5.39	<code>__gnu_cxx::encoding_char_traits< _CharT ></code> Struct Template Reference	955
5.39.1	Detailed Description	956
5.40	<code>__gnu_cxx::encoding_state</code> Class Reference	957
5.40.1	Detailed Description	958
5.41	<code>__gnu_cxx::forced_error</code> Struct Reference	958
5.41.1	Detailed Description	958
5.41.2	Member Function Documentation	959
5.42	<code>__gnu_cxx::free_list</code> Class Reference	959

5.42.1 Detailed Description	960
5.42.2 Member Function Documentation	960
5.43 <code>__gnu_cxx::hash_map< _Key, _Tp, _HashFn, _EqualKey, _Alloc ></code> Class Template Reference	961
5.43.1 Detailed Description	963
5.44 <code>__gnu_cxx::hash_multimap< _Key, _Tp, _HashFn, _EqualKey, _-</code> <code>Alloc ></code> Class Template Reference	963
5.44.1 Detailed Description	965
5.45 <code>__gnu_cxx::hash_multiset< _Value, _HashFcn, _EqualKey, _Alloc ></code> Class Template Reference	965
5.45.1 Detailed Description	967
5.46 <code>__gnu_cxx::hash_set< _Value, _HashFcn, _EqualKey, _Alloc ></code> Class Template Reference	967
5.46.1 Detailed Description	969
5.47 <code>__gnu_cxx::limit_condition</code> Struct Reference	969
5.47.1 Detailed Description	970
5.48 <code>__gnu_cxx::limit_condition::always_adjustor</code> Struct Reference	970
5.48.1 Detailed Description	970
5.49 <code>__gnu_cxx::limit_condition::limit_adjustor</code> Struct Reference	970
5.49.1 Detailed Description	971
5.50 <code>__gnu_cxx::limit_condition::never_adjustor</code> Struct Reference	971
5.50.1 Detailed Description	971
5.51 <code>__gnu_cxx::malloc_allocator< _Tp ></code> Class Template Reference	971
5.51.1 Detailed Description	972
5.52 <code>__gnu_cxx::new_allocator< _Tp ></code> Class Template Reference	973
5.52.1 Detailed Description	974
5.53 <code>__gnu_cxx::project1st< _Arg1, _Arg2 ></code> Struct Template Reference	974
5.53.1 Detailed Description	975
5.53.2 Member Typedef Documentation	975
5.54 <code>__gnu_cxx::project2nd< _Arg1, _Arg2 ></code> Struct Template Reference	976
5.54.1 Detailed Description	976
5.54.2 Member Typedef Documentation	976

5.55	<code>__gnu_cxx::random_condition</code> Struct Reference	977
5.55.1	Detailed Description	978
5.56	<code>__gnu_cxx::random_condition::always_adjustor</code> Struct Reference . . .	978
5.56.1	Detailed Description	978
5.57	<code>__gnu_cxx::random_condition::group_adjustor</code> Struct Reference . . .	978
5.57.1	Detailed Description	979
5.58	<code>__gnu_cxx::random_condition::never_adjustor</code> Struct Reference . . .	979
5.58.1	Detailed Description	979
5.59	<code>__gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc ></code> Struct Template Reference	979
5.59.1	Detailed Description	982
5.60	<code>__gnu_cxx::recursive_init_error</code> Class Reference	983
5.60.1	Detailed Description	983
5.60.2	Member Function Documentation	983
5.61	<code>__gnu_cxx::rope< _CharT, _Alloc ></code> Class Template Reference . . .	984
5.61.1	Detailed Description	990
5.62	<code>__gnu_cxx::select1st< _Pair ></code> Struct Template Reference	990
5.62.1	Detailed Description	991
5.62.2	Member Typedef Documentation	991
5.63	<code>__gnu_cxx::select2nd< _Pair ></code> Struct Template Reference	991
5.63.1	Detailed Description	992
5.63.2	Member Typedef Documentation	992
5.64	<code>__gnu_cxx::slist< _Tp, _Alloc ></code> Class Template Reference	992
5.64.1	Detailed Description	995
5.65	<code>__gnu_cxx::stdio_filebuf< _CharT, _Traits ></code> Class Template Reference	995
5.65.1	Detailed Description	1000
5.65.2	Member Typedef Documentation	1000
5.65.3	Constructor & Destructor Documentation	1001
5.65.4	Member Function Documentation	1003
5.65.5	Member Data Documentation	1026
5.66	<code>__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits ></code> Class Template Reference	1033

5.66.1 Detailed Description	1036
5.66.2 Member Typedef Documentation	1037
5.66.3 Member Function Documentation	1038
5.66.4 Member Data Documentation	1057
5.67 <code>__gnu_cxx::subtractive_rng</code> Class Reference	1061
5.67.1 Detailed Description	1061
5.67.2 Member Typedef Documentation	1062
5.67.3 Constructor & Destructor Documentation	1062
5.67.4 Member Function Documentation	1062
5.68 <code>__gnu_cxx::temporary_buffer<_ForwardIterator, _Tp></code> Struct Template Reference	1063
5.68.1 Detailed Description	1064
5.68.2 Constructor & Destructor Documentation	1064
5.68.3 Member Function Documentation	1065
5.69 <code>__gnu_cxx::throw_allocator_base<_Tp, _Cond></code> Class Template Reference	1066
5.69.1 Detailed Description	1067
5.70 <code>__gnu_cxx::throw_allocator_limit<_Tp></code> Struct Template Reference	1067
5.70.1 Detailed Description	1069
5.71 <code>__gnu_cxx::throw_allocator_random<_Tp></code> Struct Template Reference	1069
5.71.1 Detailed Description	1070
5.72 <code>__gnu_cxx::throw_value_base<_Cond></code> Struct Template Reference	1071
5.72.1 Detailed Description	1071
5.73 <code>__gnu_cxx::throw_value_limit</code> Struct Reference	1072
5.73.1 Detailed Description	1073
5.74 <code>__gnu_cxx::throw_value_random</code> Struct Reference	1073
5.74.1 Detailed Description	1075
5.75 <code>__gnu_cxx::unary_compose<_Operation1, _Operation2></code> Class Template Reference	1075
5.75.1 Detailed Description	1076
5.75.2 Member Typedef Documentation	1076
5.76 <code>__gnu_debug::_After_nth_from<_Iterator></code> Class Template Reference	1076

5.76.1 Detailed Description	1077
5.77 <code>__gnu_debug::_BeforeBeginHelper< _Sequence > Struct Template Reference</code>	1077
5.77.1 Detailed Description	1077
5.78 <code>__gnu_debug::_Equal_to< _Type > Class Template Reference</code>	1077
5.78.1 Detailed Description	1078
5.79 <code>__gnu_debug::_Not_equal_to< _Type > Class Template Reference</code> .	1078
5.79.1 Detailed Description	1078
5.80 <code>__gnu_debug::_Safe_iterator< _Iterator, _Sequence > Class Template Reference</code>	1078
5.80.1 Detailed Description	1081
5.80.2 Constructor & Destructor Documentation	1081
5.80.3 Member Function Documentation	1083
5.80.4 Member Data Documentation	1090
5.81 <code>__gnu_debug::_Safe_iterator_base Class Reference</code>	1091
5.81.1 Detailed Description	1093
5.81.2 Constructor & Destructor Documentation	1093
5.81.3 Member Function Documentation	1094
5.81.4 Member Data Documentation	1095
5.82 <code>__gnu_debug::_Safe_sequence< _Sequence > Class Template Reference</code>	1096
5.82.1 Detailed Description	1098
5.82.2 Member Function Documentation	1098
5.82.3 Member Data Documentation	1100
5.83 <code>__gnu_debug::_Safe_sequence_base Class Reference</code>	1101
5.83.1 Detailed Description	1103
5.83.2 Constructor & Destructor Documentation	1103
5.83.3 Member Function Documentation	1103
5.83.4 Member Data Documentation	1105
5.84 <code>__gnu_debug::basic_string< _CharT, _Traits, _Allocator > Class Template Reference</code>	1106
5.84.1 Detailed Description	1114

5.84.2	Member Function Documentation	1114
5.84.3	Member Data Documentation	1160
5.85	<code>__gnu_parallel::__accumulate_binop_reduct<_BinOp></code> Struct Template Reference	1161
5.85.1	Detailed Description	1162
5.86	<code>__gnu_parallel::__accumulate_selector<_It></code> Struct Template Reference	1162
5.86.1	Detailed Description	1163
5.86.2	Member Function Documentation	1163
5.86.3	Member Data Documentation	1163
5.87	<code>__gnu_parallel::__adjacent_difference_selector<_It></code> Struct Template Reference	1164
5.87.1	Detailed Description	1165
5.87.2	Member Data Documentation	1165
5.88	<code>__gnu_parallel::__adjacent_find_selector</code> Struct Reference	1165
5.88.1	Detailed Description	1166
5.88.2	Member Function Documentation	1166
5.89	<code>__gnu_parallel::__binder1st<_Operation, _FirstArgumentType, _SecondArgumentType, _ResultType></code> Class Template Reference . . .	1167
5.89.1	Detailed Description	1168
5.89.2	Member Typedef Documentation	1168
5.90	<code>__gnu_parallel::__binder2nd<_Operation, _FirstArgumentType, _SecondArgumentType, _ResultType></code> Class Template Reference . . .	1169
5.90.1	Detailed Description	1170
5.90.2	Member Typedef Documentation	1170
5.91	<code>__gnu_parallel::__count_if_selector<_It, _Diff></code> Struct Template Reference	1170
5.91.1	Detailed Description	1171
5.91.2	Member Function Documentation	1171
5.91.3	Member Data Documentation	1172
5.92	<code>__gnu_parallel::__count_selector<_It, _Diff></code> Struct Template Reference	1172
5.92.1	Detailed Description	1173

5.92.2	Member Function Documentation	1173
5.92.3	Member Data Documentation	1174
5.93	<code>__gnu_parallel::__fill_selector< _It ></code> Struct Template Reference . .	1174
5.93.1	Detailed Description	1175
5.93.2	Member Function Documentation	1175
5.93.3	Member Data Documentation	1176
5.94	<code>__gnu_parallel::__find_first_of_selector< _FIterator ></code> Struct Template Reference	1176
5.94.1	Detailed Description	1177
5.94.2	Member Function Documentation	1177
5.95	<code>__gnu_parallel::__find_if_selector</code> Struct Reference	1178
5.95.1	Detailed Description	1179
5.95.2	Member Function Documentation	1179
5.96	<code>__gnu_parallel::__for_each_selector< _It ></code> Struct Template Reference	1180
5.96.1	Detailed Description	1181
5.96.2	Member Function Documentation	1181
5.96.3	Member Data Documentation	1181
5.97	<code>__gnu_parallel::__generate_selector< _It ></code> Struct Template Reference	1181
5.97.1	Detailed Description	1182
5.97.2	Member Function Documentation	1182
5.97.3	Member Data Documentation	1183
5.98	<code>__gnu_parallel::__generic_find_selector</code> Struct Reference	1183
5.98.1	Detailed Description	1184
5.99	<code>__gnu_parallel::__generic_for_each_selector< _It ></code> Struct Template Reference	1184
5.99.1	Detailed Description	1185
5.99.2	Member Data Documentation	1186
5.100	<code>__gnu_parallel::__identity_selector< _It ></code> Struct Template Reference	1186
5.100.1	Detailed Description	1187
5.100.2	Member Function Documentation	1187
5.100.3	Member Data Documentation	1187

5.101__gnu_parallel::__inner_product_selector< _It, _It2, _Tp > Struct Template Reference	1188
5.101.1 Detailed Description	1188
5.101.2 Constructor & Destructor Documentation	1189
5.101.3 Member Function Documentation	1189
5.101.4 Member Data Documentation	1190
5.102__gnu_parallel::__max_element_reduct< _Compare, _It > Struct Template Reference	1190
5.102.1 Detailed Description	1191
5.103__gnu_parallel::__min_element_reduct< _Compare, _It > Struct Template Reference	1191
5.103.1 Detailed Description	1191
5.104__gnu_parallel::__mismatch_selector Struct Reference	1192
5.104.1 Detailed Description	1192
5.104.2 Member Function Documentation	1193
5.105__gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __- sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference	1193
5.105.1 Detailed Description	1194
5.106__gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Tem- plate Reference	1194
5.106.1 Detailed Description	1194
5.107__gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __- sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference	1195
5.107.1 Detailed Description	1195
5.108__gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Tem- plate Reference	1195
5.108.1 Detailed Description	1196
5.109__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __- sentinels, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _- Compare > Struct Template Reference	1196
5.109.1 Detailed Description	1196

5.110__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference	1197
5.110.1 Detailed Description	1197
5.111__gnu_parallel::__replace_if_selector< _It, _Op, _Tp > Struct Tem- plate Reference	1197
5.111.1 Detailed Description	1198
5.111.2 Constructor & Destructor Documentation	1198
5.111.3 Member Function Documentation	1199
5.111.4 Member Data Documentation	1199
5.112__gnu_parallel::__replace_selector< _It, _Tp > Struct Template Ref- erence	1200
5.112.1 Detailed Description	1201
5.112.2 Constructor & Destructor Documentation	1201
5.112.3 Member Function Documentation	1201
5.112.4 Member Data Documentation	1202
5.113__gnu_parallel::__transform1_selector< _It > Struct Template Refer- ence	1202
5.113.1 Detailed Description	1203
5.113.2 Member Function Documentation	1203
5.113.3 Member Data Documentation	1204
5.114__gnu_parallel::__transform2_selector< _It > Struct Template Refer- ence	1204
5.114.1 Detailed Description	1205
5.114.2 Member Function Documentation	1205
5.114.3 Member Data Documentation	1205
5.115__gnu_parallel::__unary_negate< _Predicate, argument_type > Class Template Reference	1206
5.115.1 Detailed Description	1207
5.115.2 Member Typedef Documentation	1207
5.116__gnu_parallel::__DRandomShufflingGlobalData< _RAIter > Struct Template Reference	1207
5.116.1 Detailed Description	1208
5.116.2 Constructor & Destructor Documentation	1208

5.116.3 Member Data Documentation	1209
5.117__gnu_parallel::_DRSSorterPU< _RAIter, _- RandomNumberGenerator > Struct Template Reference	1210
5.117.1 Detailed Description	1211
5.117.2 Member Data Documentation	1211
5.118__gnu_parallel::_DummyReduct Struct Reference	1212
5.118.1 Detailed Description	1213
5.119__gnu_parallel::_EqualFromLess< _T1, _T2, _Compare > Class Template Reference	1213
5.119.1 Detailed Description	1214
5.119.2 Member Typedef Documentation	1214
5.120__gnu_parallel::_EqualTo< _T1, _T2 > Struct Template Reference	1215
5.120.1 Detailed Description	1216
5.120.2 Member Typedef Documentation	1216
5.121__gnu_parallel::_GuardedIterator< _RAIter, _Compare > Class Tem- plate Reference	1216
5.121.1 Detailed Description	1217
5.121.2 Constructor & Destructor Documentation	1217
5.121.3 Member Function Documentation	1218
5.121.4 Friends And Related Function Documentation	1219
5.122__gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _- IteratorCategory > Class Template Reference	1220
5.122.1 Detailed Description	1221
5.122.2 Member Typedef Documentation	1221
5.122.3 Member Data Documentation	1221
5.123__gnu_parallel::_IteratorTriple< _Iterator1, _Iterator2, _Iterator3, _- IteratorCategory > Class Template Reference	1222
5.123.1 Detailed Description	1223
5.124__gnu_parallel::_Job< _DifferenceTp > Struct Template Reference	1223
5.124.1 Detailed Description	1223
5.124.2 Member Data Documentation	1223
5.125__gnu_parallel::_Less< _T1, _T2 > Struct Template Reference	1224
5.125.1 Detailed Description	1225

5.125.2 Member Typedef Documentation	1226
5.126__gnu_parallel::_Lexicographic< _T1, _T2, _Compare > Class Template Reference	1226
5.126.1 Detailed Description	1227
5.126.2 Member Typedef Documentation	1228
5.127__gnu_parallel::_LexicographicReverse< _T1, _T2, _Compare > Class Template Reference	1228
5.127.1 Detailed Description	1229
5.127.2 Member Typedef Documentation	1230
5.128__gnu_parallel::_LoserTree< __stable, _Tp, _Compare > Class Template Reference	1230
5.128.1 Detailed Description	1231
5.128.2 Member Function Documentation	1232
5.128.3 Member Data Documentation	1233
5.129__gnu_parallel::_LoserTree< false, _Tp, _Compare > Class Template Reference	1234
5.129.1 Detailed Description	1235
5.129.2 Member Function Documentation	1235
5.129.3 Member Data Documentation	1237
5.130__gnu_parallel::_LoserTreeBase< _Tp, _Compare > Class Template Reference	1238
5.130.1 Detailed Description	1239
5.130.2 Constructor & Destructor Documentation	1239
5.130.3 Member Function Documentation	1240
5.130.4 Member Data Documentation	1241
5.131__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser Struct Reference	1242
5.131.1 Detailed Description	1242
5.131.2 Member Data Documentation	1242
5.132__gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare > Class Template Reference	1243
5.132.1 Detailed Description	1244
5.133__gnu_parallel::_LoserTreePointer< false, _Tp, _Compare > Class Template Reference	1245

5.133.1 Detailed Description	1246
5.134__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare > Class Template Reference	1246
5.134.1 Detailed Description	1247
5.135__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser Struct Reference	1247
5.135.1 Detailed Description	1247
5.136__gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _- Compare > Class Template Reference	1248
5.136.1 Detailed Description	1249
5.137__gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare > Class Template Reference	1249
5.137.1 Detailed Description	1250
5.138__gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare > Class Template Reference	1250
5.138.1 Detailed Description	1251
5.139__gnu_parallel::_LoserTreeTraits< _Tp > Struct Template Reference	1251
5.139.1 Detailed Description	1251
5.139.2 Member Data Documentation	1252
5.140__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare > Class Template Reference	1252
5.140.1 Detailed Description	1253
5.141__gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare > Class Template Reference	1254
5.141.1 Detailed Description	1255
5.142__gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare > Class Template Reference	1255
5.142.1 Detailed Description	1256
5.143__gnu_parallel::_Multiplies< _Tp1, _Tp2, _Result > Struct Template Reference	1256
5.143.1 Detailed Description	1257
5.143.2 Member Typedef Documentation	1258
5.144__gnu_parallel::_Nothing Struct Reference	1258
5.144.1 Detailed Description	1258

5.144.2 Member Function Documentation	1259
5.145 __gnu_parallel::_Piece< _DifferenceTp > Struct Template Reference	1259
5.145.1 Detailed Description	1259
5.145.2 Member Data Documentation	1260
5.146 __gnu_parallel::_Plus< _Tp1, _Tp2, _Result > Struct Template Reference	1260
5.146.1 Detailed Description	1261
5.146.2 Member Typedef Documentation	1262
5.147 __gnu_parallel::_PMWMSortingData< _RAIter > Struct Template Reference	1262
5.147.1 Detailed Description	1263
5.147.2 Member Data Documentation	1263
5.148 __gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp > Class Template Reference	1265
5.148.1 Detailed Description	1265
5.148.2 Constructor & Destructor Documentation	1266
5.148.3 Member Function Documentation	1266
5.149 __gnu_parallel::_PseudoSequenceIterator< _Tp, _DifferenceTp > Class Template Reference	1267
5.149.1 Detailed Description	1267
5.150 __gnu_parallel::_QSBThreadLocal< _RAIter > Struct Template Reference	1267
5.150.1 Detailed Description	1268
5.150.2 Member Typedef Documentation	1268
5.150.3 Constructor & Destructor Documentation	1268
5.150.4 Member Data Documentation	1269
5.151 __gnu_parallel::_RandomNumber Class Reference	1270
5.151.1 Detailed Description	1270
5.151.2 Constructor & Destructor Documentation	1271
5.151.3 Member Function Documentation	1271
5.152 __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp > Class Template Reference	1272
5.152.1 Detailed Description	1273

5.152.2 Constructor & Destructor Documentation	1273
5.152.3 Member Function Documentation	1274
5.153 <code>__gnu_parallel::_SamplingSorter< __stable, _RAIter, _-</code> <code>StrictWeakOrdering ></code> Struct Template Reference	1275
5.153.1 Detailed Description	1275
5.154 <code>__gnu_parallel::_SamplingSorter< false, _RAIter, _-</code> <code>StrictWeakOrdering ></code> Struct Template Reference	1275
5.154.1 Detailed Description	1275
5.155 <code>__gnu_parallel::_Settings</code> Struct Reference	1276
5.155.1 Detailed Description	1277
5.155.2 Member Function Documentation	1277
5.155.3 Member Data Documentation	1278
5.156 <code>__gnu_parallel::_SplitConsistently< __exact, _RAIter, _Compare, _-</code> <code>SortingPlacesIterator ></code> Struct Template Reference	1286
5.156.1 Detailed Description	1286
5.157 <code>__gnu_parallel::_SplitConsistently< false, _RAIter, _Compare, _-</code> <code>SortingPlacesIterator ></code> Struct Template Reference	1287
5.157.1 Detailed Description	1287
5.158 <code>__gnu_parallel::_SplitConsistently< true, _RAIter, _Compare, _-</code> <code>SortingPlacesIterator ></code> Struct Template Reference	1287
5.158.1 Detailed Description	1288
5.159 <code>__gnu_parallel::balanced_quicksort_tag</code> Struct Reference	1288
5.159.1 Detailed Description	1288
5.159.2 Member Function Documentation	1289
5.160 <code>__gnu_parallel::balanced_tag</code> Struct Reference	1289
5.160.1 Detailed Description	1290
5.160.2 Member Function Documentation	1290
5.161 <code>__gnu_parallel::constant_size_blocks_tag</code> Struct Reference	1291
5.161.1 Detailed Description	1291
5.162 <code>__gnu_parallel::default_parallel_tag</code> Struct Reference	1292
5.162.1 Detailed Description	1292
5.162.2 Member Function Documentation	1293
5.163 <code>__gnu_parallel::equal_split_tag</code> Struct Reference	1293

5.163.1 Detailed Description	1294
5.164__gnu_parallel::exact_tag Struct Reference	1294
5.164.1 Detailed Description	1295
5.164.2 Member Function Documentation	1295
5.165__gnu_parallel::find_tag Struct Reference	1296
5.165.1 Detailed Description	1296
5.166__gnu_parallel::growing_blocks_tag Struct Reference	1297
5.166.1 Detailed Description	1297
5.167__gnu_parallel::multiway_mergesort_exact_tag Struct Reference	1297
5.167.1 Detailed Description	1298
5.167.2 Member Function Documentation	1298
5.168__gnu_parallel::multiway_mergesort_sampling_tag Struct Reference	1299
5.168.1 Detailed Description	1300
5.168.2 Member Function Documentation	1300
5.169__gnu_parallel::multiway_mergesort_tag Struct Reference	1300
5.169.1 Detailed Description	1301
5.169.2 Member Function Documentation	1301
5.170__gnu_parallel::omp_loop_static_tag Struct Reference	1302
5.170.1 Detailed Description	1302
5.170.2 Member Function Documentation	1303
5.171__gnu_parallel::omp_loop_tag Struct Reference	1303
5.171.1 Detailed Description	1304
5.171.2 Member Function Documentation	1304
5.172__gnu_parallel::parallel_tag Struct Reference	1305
5.172.1 Detailed Description	1307
5.172.2 Constructor & Destructor Documentation	1307
5.172.3 Member Function Documentation	1307
5.173__gnu_parallel::quicksort_tag Struct Reference	1308
5.173.1 Detailed Description	1309
5.173.2 Member Function Documentation	1309
5.174__gnu_parallel::sampling_tag Struct Reference	1309

5.174.1 Detailed Description	1310
5.174.2 Member Function Documentation	1310
5.175 <code>__gnu_parallel::sequential_tag</code> Struct Reference	1311
5.175.1 Detailed Description	1311
5.176 <code>__gnu_parallel::unbalanced_tag</code> Struct Reference	1311
5.176.1 Detailed Description	1312
5.176.2 Member Function Documentation	1312
5.177 <code>__gnu_pbds::associative_container_tag</code> Struct Reference	1313
5.177.1 Detailed Description	1313
5.178 <code>__gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, Allocator > Class Template Reference</code>	1314
5.178.1 Detailed Description	1315
5.179 <code>__gnu_pbds::basic_hash_tag</code> Struct Reference	1315
5.179.1 Detailed Description	1316
5.180 <code>__gnu_pbds::basic_tree< Key, Mapped, Tag, Node_Update, Policy_Tl, Allocator > Class Template Reference</code>	1316
5.180.1 Detailed Description	1317
5.181 <code>__gnu_pbds::basic_tree_tag</code> Struct Reference	1317
5.181.1 Detailed Description	1317
5.182 <code>__gnu_pbds::binary_heap_tag</code> Struct Reference	1318
5.182.1 Detailed Description	1318
5.183 <code>__gnu_pbds::binomial_heap_tag</code> Struct Reference	1318
5.183.1 Detailed Description	1319
5.184 <code>__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, Allocator > Class Template Reference</code>	1319
5.184.1 Detailed Description	1321
5.185 <code>__gnu_pbds::cc_hash_tag</code> Struct Reference	1321
5.185.1 Detailed Description	1322
5.186 <code>__gnu_pbds::container_base< Key, Mapped, Tag, Policy_Tl, Allocator > Class Template Reference</code>	1322
5.186.1 Detailed Description	1323

5.187__gnu_pbds::container_tag Struct Reference	1324
5.187.1 Detailed Description	1324
5.188__gnu_pbds::container_traits< Cntnr > Struct Template Reference . .	1324
5.188.1 Detailed Description	1325
5.189__gnu_pbds::detail::value_type_base< Key, Mapped, Allocator, false > Struct Template Reference	1325
5.189.1 Detailed Description	1326
5.190__gnu_pbds::detail::value_type_base< Key, Mapped, Allocator, true > Struct Template Reference	1326
5.190.1 Detailed Description	1326
5.191__gnu_pbds::detail::value_type_base< Key, null_mapped_type, Allo- cator, false > Struct Template Reference	1327
5.191.1 Detailed Description	1327
5.192__gnu_pbds::detail::value_type_base< Key, null_mapped_type, Allo- cator, true > Struct Template Reference	1328
5.192.1 Detailed Description	1328
5.193__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, Allocator > Class Template Reference	1328
5.193.1 Detailed Description	1330
5.194__gnu_pbds::gp_hash_tag Struct Reference	1331
5.194.1 Detailed Description	1331
5.195__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, Al- locator > Class Template Reference	1332
5.195.1 Detailed Description	1333
5.196__gnu_pbds::list_update_tag Struct Reference	1333
5.196.1 Detailed Description	1334
5.197__gnu_pbds::null_mapped_type Struct Reference	1334
5.197.1 Detailed Description	1334
5.198__gnu_pbds::ov_tree_tag Struct Reference	1335
5.198.1 Detailed Description	1335
5.199__gnu_pbds::pairing_heap_tag Struct Reference	1336
5.199.1 Detailed Description	1336

5.200 <code>__gnu_pbds::pat_trie_tag</code> Struct Reference	1336
5.200.1 Detailed Description	1337
5.201 <code>__gnu_pbds::priority_queue_tag</code> Struct Reference	1337
5.201.1 Detailed Description	1338
5.202 <code>__gnu_pbds::rb_tree_tag</code> Struct Reference	1338
5.202.1 Detailed Description	1339
5.203 <code>__gnu_pbds::rc_binomial_heap_tag</code> Struct Reference	1339
5.203.1 Detailed Description	1340
5.204 <code>__gnu_pbds::sequence_tag</code> Struct Reference	1340
5.204.1 Detailed Description	1341
5.205 <code>__gnu_pbds::splay_tree_tag</code> Struct Reference	1341
5.205.1 Detailed Description	1342
5.206 <code>__gnu_pbds::string_tag</code> Struct Reference	1342
5.206.1 Detailed Description	1343
5.207 <code>__gnu_pbds::thin_heap_tag</code> Struct Reference	1343
5.207.1 Detailed Description	1344
5.208 <code>__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, Allo-</code> <code>cator ></code> Class Template Reference	1344
5.208.1 Detailed Description	1346
5.209 <code>__gnu_pbds::tree_tag</code> Struct Reference	1346
5.209.1 Detailed Description	1347
5.210 <code>__gnu_pbds::trie< Key, Mapped, E_Access_Traits, Tag, Node_-</code> <code>Update, Allocator ></code> Class Template Reference	1347
5.210.1 Detailed Description	1348
5.211 <code>__gnu_pbds::trie_tag</code> Struct Reference	1348
5.211.1 Detailed Description	1349
5.212 <code>__gnu_profile::__container_size_info</code> Class Reference	1349
5.212.1 Detailed Description	1351
5.213 <code>__gnu_profile::__container_size_stack_info</code> Class Reference	1351
5.213.1 Detailed Description	1352
5.214 <code>__gnu_profile::__hashfunc_info</code> Class Reference	1352
5.214.1 Detailed Description	1353

5.215 <code>__gnu_profile::__hashfunc_stack_info</code> Class Reference	1353
5.215.1 Detailed Description	1354
5.216 <code>__gnu_profile::__list2vector_info</code> Class Reference	1355
5.216.1 Detailed Description	1356
5.217 <code>__gnu_profile::__map2umap_info</code> Class Reference	1356
5.217.1 Detailed Description	1357
5.218 <code>__gnu_profile::__map2umap_stack_info</code> Class Reference	1357
5.218.1 Detailed Description	1359
5.219 <code>__gnu_profile::__object_info_base</code> Class Reference	1359
5.219.1 Detailed Description	1360
5.220 <code>__gnu_profile::__reentrance_guard</code> Struct Reference	1360
5.220.1 Detailed Description	1360
5.221 <code>__gnu_profile::__stack_hash</code> Class Reference	1360
5.221.1 Detailed Description	1360
5.222 <code>__gnu_profile::__stack_info_base< __object_info ></code> Class Template Reference	1361
5.222.1 Detailed Description	1361
5.223 <code>__gnu_profile::__trace_base< __object_info, __stack_info ></code> Class Template Reference	1361
5.223.1 Detailed Description	1362
5.224 <code>__gnu_profile::__trace_container_size</code> Class Reference	1362
5.224.1 Detailed Description	1363
5.225 <code>__gnu_profile::__trace_hash_func</code> Class Reference	1363
5.225.1 Detailed Description	1364
5.226 <code>__gnu_profile::__trace_hashtable_size</code> Class Reference	1364
5.226.1 Detailed Description	1365
5.227 <code>__gnu_profile::__trace_map2umap</code> Class Reference	1365
5.227.1 Detailed Description	1366
5.228 <code>__gnu_profile::__trace_vector_size</code> Class Reference	1366
5.228.1 Detailed Description	1367
5.229 <code>__gnu_profile::__trace_vector_to_list</code> Class Reference	1367
5.229.1 Detailed Description	1368

5.230__gnu_profile::__vector2list_info Class Reference	1368
5.230.1 Detailed Description	1370
5.231__gnu_profile::__vector2list_stack_info Class Reference	1370
5.231.1 Detailed Description	1371
5.232__gnu_profile::__warning_data Struct Reference	1371
5.232.1 Detailed Description	1372
5.233std::__atomic0::__atomic_base< _ITp > Struct Template Reference .	1372
5.233.1 Detailed Description	1374
5.234std::__atomic0::atomic_address Struct Reference	1374
5.234.1 Detailed Description	1376
5.235std::__atomic0::atomic_flag Struct Reference	1376
5.235.1 Detailed Description	1377
5.236std::__atomic2::__atomic_base< _ITp > Struct Template Reference .	1377
5.236.1 Detailed Description	1379
5.237std::__atomic2::atomic_address Struct Reference	1379
5.237.1 Detailed Description	1381
5.238std::__atomic2::atomic_flag Struct Reference	1381
5.238.1 Detailed Description	1382
5.239std::__atomic_flag_base Struct Reference	1382
5.239.1 Detailed Description	1382
5.240std::__basic_future< _Res > Class Template Reference	1383
5.240.1 Detailed Description	1384
5.240.2 Member Function Documentation	1384
5.241std::__codecvt_abstract_base< _InternT, _ExternT, _StateT > Class Template Reference	1384
5.241.1 Detailed Description	1386
5.241.2 Member Function Documentation	1386
5.242std::__ctype_abstract_base< _CharT > Class Template Reference . .	1390
5.242.1 Detailed Description	1392
5.242.2 Member Typedef Documentation	1393
5.242.3 Member Function Documentation	1393

5.243std::__debug::bitset< _Nb > Class Template Reference	1406
5.243.1 Detailed Description	1407
5.244std::__debug::deque< _Tp, _Allocator > Class Template Reference . .	1408
5.244.1 Detailed Description	1411
5.244.2 Member Function Documentation	1411
5.244.3 Member Data Documentation	1413
5.245std::__debug::forward_list< _Tp, _Alloc > Class Template Reference	1414
5.245.1 Detailed Description	1417
5.245.2 Member Function Documentation	1417
5.245.3 Member Data Documentation	1419
5.246std::__debug::list< _Tp, _Allocator > Class Template Reference . . .	1420
5.246.1 Detailed Description	1424
5.246.2 Member Function Documentation	1424
5.246.3 Member Data Documentation	1426
5.247std::__debug::map< _Key, _Tp, _Compare, _Allocator > Class Tem- plate Reference	1427
5.247.1 Detailed Description	1430
5.247.2 Member Function Documentation	1430
5.247.3 Member Data Documentation	1432
5.248std::__debug::multimap< _Key, _Tp, _Compare, _Allocator > Class Template Reference	1433
5.248.1 Detailed Description	1436
5.248.2 Member Function Documentation	1436
5.248.3 Member Data Documentation	1438
5.249std::__debug::multiset< _Key, _Compare, _Allocator > Class Tem- plate Reference	1439
5.249.1 Detailed Description	1442
5.249.2 Member Function Documentation	1442
5.249.3 Member Data Documentation	1444
5.250std::__debug::set< _Key, _Compare, _Allocator > Class Template Reference	1445
5.250.1 Detailed Description	1448

5.250.2 Member Function Documentation	1449
5.250.3 Member Data Documentation	1451
5.251std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference	1452
5.251.1 Detailed Description	1454
5.251.2 Member Function Documentation	1454
5.251.3 Member Data Documentation	1457
5.252std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference	1457
5.252.1 Detailed Description	1460
5.252.2 Member Function Documentation	1460
5.252.3 Member Data Documentation	1462
5.253std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc > Class Template Reference	1463
5.253.1 Detailed Description	1465
5.253.2 Member Function Documentation	1465
5.253.3 Member Data Documentation	1467
5.254std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc > Class Template Reference	1468
5.254.1 Detailed Description	1471
5.254.2 Member Function Documentation	1471
5.254.3 Member Data Documentation	1473
5.255std::__debug::vector< _Tp, _Allocator > Class Template Reference	1474
5.255.1 Detailed Description	1478
5.255.2 Constructor & Destructor Documentation	1478
5.255.3 Member Function Documentation	1478
5.255.4 Member Data Documentation	1480
5.256std::__declval_protector< _Tp > Struct Template Reference	1481
5.256.1 Detailed Description	1481
5.257std::__detail::_List_node_base Struct Reference	1482
5.257.1 Detailed Description	1482
5.258std::__exception_ptr::exception_ptr Class Reference	1483

5.258.1 Detailed Description	1483
5.259std::__future_base Struct Reference	1483
5.259.1 Detailed Description	1485
5.260std::__future_base::_Ptr< _Res > Struct Template Reference	1485
5.260.1 Detailed Description	1485
5.261std::__future_base::_Result< _Res > Struct Template Reference	1486
5.261.1 Detailed Description	1486
5.262std::__future_base::_Result< _Res & > Struct Template Reference	1487
5.262.1 Detailed Description	1487
5.263std::__future_base::_Result< void > Struct Template Reference	1488
5.263.1 Detailed Description	1488
5.264std::__future_base::_Result_alloc< _Res, _Alloc > Struct Template Reference	1488
5.264.1 Detailed Description	1489
5.265std::__future_base::_Result_base Struct Reference	1490
5.265.1 Detailed Description	1490
5.266std::__future_base::_State Class Reference	1491
5.266.1 Detailed Description	1491
5.267std::__has_iterator_category_helper< _Tp > Class Template Reference	1492
5.267.1 Detailed Description	1492
5.268std::__is_location_invariant< _Tp > Struct Template Reference	1492
5.268.1 Detailed Description	1493
5.269std::__is_member_pointer_helper< _Tp > Struct Template Reference	1493
5.269.1 Detailed Description	1494
5.270std::__numeric_limits_base Struct Reference	1494
5.270.1 Detailed Description	1495
5.270.2 Member Data Documentation	1496
5.271std::__parallel::__CRandNumber< _MustBeInt > Struct Template Ref- erence	1499
5.271.1 Detailed Description	1500
5.272std::__profile::bitset< _Nb > Class Template Reference	1500
5.272.1 Detailed Description	1501

5.273std::__profile::deque< _Tp, _Allocator > Class Template Reference .	1502
5.273.1 Detailed Description	1504
5.274std::__profile::forward_list< _Tp, _Alloc > Class Template Reference	1504
5.274.1 Detailed Description	1505
5.275std::__profile::list< _Tp, _Allocator > Class Template Reference . .	1505
5.275.1 Detailed Description	1507
5.276std::__profile::map< _Key, _Tp, _Compare, _Allocator > Class Tem- plate Reference	1508
5.276.1 Detailed Description	1510
5.277std::__profile::multimap< _Key, _Tp, _Compare, _Allocator > Class Template Reference	1510
5.277.1 Detailed Description	1512
5.278std::__profile::multiset< _Key, _Compare, _Allocator > Class Tem- plate Reference	1512
5.278.1 Detailed Description	1514
5.279std::__profile::set< _Key, _Compare, _Allocator > Class Template Reference	1514
5.279.1 Detailed Description	1516
5.280std::__profile::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference	1516
5.280.1 Detailed Description	1518
5.281std::__profile::unordered_multimap< _Key, _Tp, _Hash, _Pred, _- Alloc > Class Template Reference	1518
5.281.1 Detailed Description	1519
5.282std::__profile::unordered_multiset< _Value, _Hash, _Pred, _Alloc > Class Template Reference	1520
5.282.1 Detailed Description	1521
5.283std::__profile::unordered_set< _Key, _Hash, _Pred, _Alloc > Class Template Reference	1521
5.283.1 Detailed Description	1522
5.284std::_Base_bitset< _Nw > Struct Template Reference	1523
5.284.1 Detailed Description	1524
5.284.2 Member Data Documentation	1524
5.285std::_Base_bitset< 0 > Struct Template Reference	1524

5.285.1 Detailed Description	1525
5.286std::_Base_bitset< 1 > Struct Template Reference	1525
5.286.1 Detailed Description	1527
5.287std::_Build_index_tuple< _Num > Struct Template Reference	1527
5.287.1 Detailed Description	1527
5.288std::_Deque_base< _Tp, _Alloc > Class Template Reference	1528
5.288.1 Detailed Description	1529
5.288.2 Member Function Documentation	1529
5.289std::_Deque_iterator< _Tp, _Ref, _Ptr > Struct Template Reference	1530
5.289.1 Detailed Description	1531
5.289.2 Member Function Documentation	1532
5.290std::_Derives_from_binary_function< _Tp > Struct Template Reference	1532
5.290.1 Detailed Description	1532
5.291std::_Derives_from_unary_function< _Tp > Struct Template Reference	1533
5.291.1 Detailed Description	1533
5.292std::_Function_base Class Reference	1533
5.292.1 Detailed Description	1534
5.293std::_Function_to_function_pointer< _Tp, _IsFunctionType > Struct Template Reference	1535
5.293.1 Detailed Description	1535
5.294std::_Fwd_list_base< _Tp, _Alloc > Struct Template Reference	1535
5.294.1 Detailed Description	1537
5.295std::_Fwd_list_const_iterator< _Tp > Struct Template Reference	1537
5.295.1 Detailed Description	1538
5.296std::_Fwd_list_iterator< _Tp > Struct Template Reference	1538
5.296.1 Detailed Description	1539
5.297std::_Fwd_list_node< _Tp > Struct Template Reference	1539
5.297.1 Detailed Description	1540
5.298std::_Fwd_list_node_base Struct Reference	1541
5.298.1 Detailed Description	1541
5.299std::_Index_tuple< _Indexes > Struct Template Reference	1542

5.299.1 Detailed Description	1542
5.300std::_List_base< _Tp, _Alloc > Class Template Reference	1542
5.300.1 Detailed Description	1543
5.301std::_List_const_iterator< _Tp > Struct Template Reference	1543
5.301.1 Detailed Description	1544
5.302std::_List_iterator< _Tp > Struct Template Reference	1544
5.302.1 Detailed Description	1545
5.303std::_List_node< _Tp > Struct Template Reference	1545
5.303.1 Detailed Description	1546
5.303.2 Member Data Documentation	1547
5.304std::_Maybe_get_result_type< _Has_result_type, _Functor > Struct Template Reference	1547
5.304.1 Detailed Description	1547
5.305std::_Maybe_unary_or_binary_function< _Res, _ArgTypes > Struct Template Reference	1548
5.305.1 Detailed Description	1548
5.306std::_Maybe_unary_or_binary_function< _Res, _T1 > Struct Tem- plate Reference	1548
5.306.1 Detailed Description	1549
5.306.2 Member Typedef Documentation	1549
5.307std::_Maybe_unary_or_binary_function< _Res, _T1, _T2 > Struct Template Reference	1550
5.307.1 Detailed Description	1551
5.307.2 Member Typedef Documentation	1551
5.308std::_Maybe_wrap_member_pointer< _Tp > Struct Template Reference	1552
5.308.1 Detailed Description	1552
5.309std::_Maybe_wrap_member_pointer< _Tp _Class::* > Struct Tem- plate Reference	1552
5.309.1 Detailed Description	1552
5.310std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) const > Class Tem- plate Reference	1553
5.310.1 Detailed Description	1554

5.311	<code>std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) const volatile > Class</code> Template Reference	1554
5.311.1	Detailed Description	1554
5.312	<code>std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) volatile > Class</code> Tem- plate Reference	1555
5.312.1	Detailed Description	1556
5.313	<code>std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) > Class</code> Template Ref- erence	1556
5.313.1	Detailed Description	1557
5.314	<code>std::_Mu< _Arg, false, false > Class</code> Template Reference	1557
5.314.1	Detailed Description	1557
5.315	<code>std::_Mu< _Arg, false, true > Class</code> Template Reference	1558
5.315.1	Detailed Description	1558
5.316	<code>std::_Mu< _Arg, true, false > Class</code> Template Reference	1558
5.316.1	Detailed Description	1558
5.317	<code>std::_Mu< reference_wrapper< _Tp >, false, false > Class</code> Template Reference	1559
5.317.1	Detailed Description	1559
5.318	<code>std::_Placeholder< _Num > Struct</code> Template Reference	1559
5.318.1	Detailed Description	1559
5.319	<code>std::_Reference_wrapper_base< _Tp > Struct</code> Template Reference	1560
5.319.1	Detailed Description	1560
5.320	<code>std::_Safe_tuple_element< __i, _Tuple > Struct</code> Template Reference	1561
5.320.1	Detailed Description	1561
5.321	<code>std::_Safe_tuple_element_impl< __i, _Tuple, _IsSafe > Struct</code> Tem- plate Reference	1561
5.321.1	Detailed Description	1561
5.322	<code>std::_Safe_tuple_element_impl< __i, _Tuple, false > Struct</code> Template Reference	1562
5.322.1	Detailed Description	1562
5.323	<code>std::_Temporary_buffer< _ForwardIterator, _Tp > Class</code> Template Reference	1563
5.323.1	Detailed Description	1564

5.323.2 Constructor & Destructor Documentation	1564
5.323.3 Member Function Documentation	1564
5.324std::_Tuple_impl<_Idx> Struct Template Reference	1565
5.324.1 Detailed Description	1565
5.325std::_Tuple_impl<_Idx, _Head, _Tail...> Struct Template Reference	1566
5.325.1 Detailed Description	1567
5.326std::_Vector_base<_Tp, _Alloc> Struct Template Reference	1567
5.326.1 Detailed Description	1568
5.327std::_Weak_result_type<_Functor> Struct Template Reference	1568
5.327.1 Detailed Description	1568
5.328std::_Weak_result_type_impl<_Functor> Struct Template Reference	1569
5.328.1 Detailed Description	1569
5.329std::_Weak_result_type_impl<_Res(&)(_ArgTypes...)> Struct Template Reference	1569
5.329.1 Detailed Description	1569
5.330std::_Weak_result_type_impl<_Res(*)(_ArgTypes...)> Struct Template Reference	1570
5.330.1 Detailed Description	1570
5.331std::_Weak_result_type_impl<_Res(_ArgTypes...)> Struct Template Reference	1570
5.331.1 Detailed Description	1570
5.332std::_Weak_result_type_impl<_Res(_Class::*)(_ArgTypes...) const> Struct Template Reference	1571
5.332.1 Detailed Description	1571
5.333std::_Weak_result_type_impl<_Res(_Class::*)(_ArgTypes...) const volatile> Struct Template Reference	1571
5.333.1 Detailed Description	1571
5.334std::_Weak_result_type_impl<_Res(_Class::*)(_ArgTypes...) volatile> Struct Template Reference	1572
5.334.1 Detailed Description	1572
5.335std::_Weak_result_type_impl<_Res(_Class::*)(_ArgTypes...)> Struct Template Reference	1572
5.335.1 Detailed Description	1572

5.336std::add_const< _Tp > Struct Template Reference	1573
5.336.1 Detailed Description	1573
5.337std::add_cv< _Tp > Struct Template Reference	1573
5.337.1 Detailed Description	1573
5.338std::add_lvalue_reference< _Tp > Struct Template Reference	1573
5.338.1 Detailed Description	1574
5.339std::add_pointer< _Tp > Struct Template Reference	1574
5.339.1 Detailed Description	1574
5.340std::add_rvalue_reference< _Tp > Struct Template Reference	1574
5.340.1 Detailed Description	1575
5.341std::add_volatile< _Tp > Struct Template Reference	1575
5.341.1 Detailed Description	1575
5.342std::adopt_lock_t Struct Reference	1575
5.342.1 Detailed Description	1575
5.343std::aligned_storage< _Len, _Align > Struct Template Reference	1576
5.343.1 Detailed Description	1576
5.344std::alignment_of< _Tp > Struct Template Reference	1577
5.344.1 Detailed Description	1578
5.345std::allocator< _Tp > Class Template Reference	1578
5.345.1 Detailed Description	1579
5.346std::allocator< void > Class Template Reference	1579
5.346.1 Detailed Description	1580
5.347std::allocator_arg_t Struct Reference	1580
5.347.1 Detailed Description	1580
5.348std::array< _Tp, _Nm > Struct Template Reference	1580
5.348.1 Detailed Description	1581
5.349std::atomic< _Tp > Struct Template Reference	1582
5.349.1 Detailed Description	1583
5.350std::atomic< _Tp * > Struct Template Reference	1583
5.350.1 Detailed Description	1584
5.351std::atomic< bool > Struct Template Reference	1585

5.351.1 Detailed Description	1586
5.352std::atomic< char > Struct Template Reference	1586
5.352.1 Detailed Description	1587
5.353std::atomic< char16_t > Struct Template Reference	1587
5.353.1 Detailed Description	1587
5.354std::atomic< char32_t > Struct Template Reference	1588
5.354.1 Detailed Description	1588
5.355std::atomic< int > Struct Template Reference	1588
5.355.1 Detailed Description	1589
5.356std::atomic< long > Struct Template Reference	1589
5.356.1 Detailed Description	1589
5.357std::atomic< long long > Struct Template Reference	1590
5.357.1 Detailed Description	1590
5.358std::atomic< short > Struct Template Reference	1590
5.358.1 Detailed Description	1591
5.359std::atomic< signed char > Struct Template Reference	1591
5.359.1 Detailed Description	1591
5.360std::atomic< unsigned char > Struct Template Reference	1592
5.360.1 Detailed Description	1592
5.361std::atomic< unsigned int > Struct Template Reference	1592
5.361.1 Detailed Description	1593
5.362std::atomic< unsigned long > Struct Template Reference	1593
5.362.1 Detailed Description	1593
5.363std::atomic< unsigned long long > Struct Template Reference	1594
5.363.1 Detailed Description	1594
5.364std::atomic< unsigned short > Struct Template Reference	1594
5.364.1 Detailed Description	1595
5.365std::atomic< wchar_t > Struct Template Reference	1595
5.365.1 Detailed Description	1596
5.366std::atomic_bool Struct Reference	1596
5.366.1 Detailed Description	1597

5.367	<code>std::auto_ptr< _Tp ></code> Class Template Reference	1597
5.367.1	Detailed Description	1598
5.367.2	Member Typedef Documentation	1599
5.367.3	Constructor & Destructor Documentation	1599
5.367.4	Member Function Documentation	1601
5.368	<code>std::auto_ptr_ref< _Tp1 ></code> Struct Template Reference	1603
5.368.1	Detailed Description	1603
5.369	<code>std::back_insert_iterator< _Container ></code> Class Template Reference . .	1604
5.369.1	Detailed Description	1605
5.369.2	Member Typedef Documentation	1605
5.369.3	Constructor & Destructor Documentation	1606
5.369.4	Member Function Documentation	1607
5.370	<code>std::bad_alloc</code> Class Reference	1608
5.370.1	Detailed Description	1608
5.370.2	Member Function Documentation	1609
5.371	<code>std::bad_cast</code> Class Reference	1609
5.371.1	Detailed Description	1609
5.371.2	Member Function Documentation	1610
5.372	<code>std::bad_exception</code> Class Reference	1610
5.372.1	Detailed Description	1611
5.372.2	Member Function Documentation	1611
5.373	<code>std::bad_function_call</code> Class Reference	1611
5.373.1	Detailed Description	1612
5.373.2	Member Function Documentation	1612
5.374	<code>std::bad_typeid</code> Class Reference	1612
5.374.1	Detailed Description	1613
5.374.2	Member Function Documentation	1613
5.375	<code>std::bad_weak_ptr</code> Class Reference	1613
5.375.1	Detailed Description	1614
5.375.2	Member Function Documentation	1614
5.376	<code>std::basic_filebuf< _CharT, _Traits ></code> Class Template Reference . . .	1614

5.376.1 Detailed Description	1618
5.376.2 Member Typedef Documentation	1618
5.376.3 Constructor & Destructor Documentation	1620
5.376.4 Member Function Documentation	1620
5.376.5 Member Data Documentation	1643
5.377std::basic_fstream< _CharT, _Traits > Class Template Reference . .	1649
5.377.1 Detailed Description	1657
5.377.2 Member Typedef Documentation	1658
5.377.3 Member Enumeration Documentation	1664
5.377.4 Constructor & Destructor Documentation	1664
5.377.5 Member Function Documentation	1666
5.377.6 Member Data Documentation	1715
5.378std::basic_ifstream< _CharT, _Traits > Class Template Reference . .	1723
5.378.1 Detailed Description	1730
5.378.2 Member Typedef Documentation	1730
5.378.3 Member Enumeration Documentation	1735
5.378.4 Constructor & Destructor Documentation	1735
5.378.5 Member Function Documentation	1736
5.378.6 Member Data Documentation	1774
5.379std::basic_ios< _CharT, _Traits > Class Template Reference	1783
5.379.1 Detailed Description	1787
5.379.2 Member Typedef Documentation	1787
5.379.3 Member Enumeration Documentation	1792
5.379.4 Constructor & Destructor Documentation	1793
5.379.5 Member Function Documentation	1793
5.379.6 Member Data Documentation	1810
5.380std::basic_iostream< _CharT, _Traits > Class Template Reference . .	1819
5.380.1 Detailed Description	1826
5.380.2 Member Typedef Documentation	1826
5.380.3 Member Enumeration Documentation	1833
5.380.4 Constructor & Destructor Documentation	1833

5.380.5 Member Function Documentation	1834
5.380.6 Member Data Documentation	1881
5.381std::basic_istream< _CharT, _Traits > Class Template Reference . .	1890
5.381.1 Detailed Description	1896
5.381.2 Member Typedef Documentation	1896
5.381.3 Member Enumeration Documentation	1901
5.381.4 Constructor & Destructor Documentation	1901
5.381.5 Member Function Documentation	1902
5.381.6 Member Data Documentation	1938
5.382std::basic_istream< _CharT, _Traits >::sentry Class Reference	1947
5.382.1 Detailed Description	1947
5.382.2 Member Typedef Documentation	1948
5.382.3 Constructor & Destructor Documentation	1948
5.382.4 Member Function Documentation	1949
5.383std::basic_istream< _CharT, _Traits, _Alloc > Class Template Reference	1949
5.383.1 Detailed Description	1956
5.383.2 Member Typedef Documentation	1956
5.383.3 Member Enumeration Documentation	1961
5.383.4 Constructor & Destructor Documentation	1961
5.383.5 Member Function Documentation	1962
5.383.6 Member Data Documentation	1999
5.384std::basic_ofstream< _CharT, _Traits > Class Template Reference . .	2008
5.384.1 Detailed Description	2014
5.384.2 Member Typedef Documentation	2014
5.384.3 Member Enumeration Documentation	2019
5.384.4 Constructor & Destructor Documentation	2019
5.384.5 Member Function Documentation	2021
5.384.6 Member Data Documentation	2050
5.385std::basic_ostream< _CharT, _Traits > Class Template Reference . .	2058
5.385.1 Detailed Description	2063

5.385.2 Member Typedef Documentation	2064
5.385.3 Member Enumeration Documentation	2069
5.385.4 Constructor & Destructor Documentation	2069
5.385.5 Member Function Documentation	2069
5.385.6 Member Data Documentation	2097
5.386std::basic_ostream< _CharT, _Traits >::sentry Class Reference . . .	2105
5.386.1 Detailed Description	2106
5.386.2 Constructor & Destructor Documentation	2106
5.386.3 Member Function Documentation	2107
5.387std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template Reference	2107
5.387.1 Detailed Description	2113
5.387.2 Member Typedef Documentation	2113
5.387.3 Member Enumeration Documentation	2118
5.387.4 Constructor & Destructor Documentation	2118
5.387.5 Member Function Documentation	2119
5.387.6 Member Data Documentation	2148
5.388std::basic_regex< _Ch_type, _Rx_traits > Class Template Reference	2156
5.388.1 Detailed Description	2157
5.388.2 Constructor & Destructor Documentation	2158
5.388.3 Member Function Documentation	2161
5.389std::basic_streambuf< _CharT, _Traits > Class Template Reference .	2168
5.389.1 Detailed Description	2170
5.389.2 Member Typedef Documentation	2172
5.389.3 Constructor & Destructor Documentation	2174
5.389.4 Member Function Documentation	2174
5.389.5 Member Data Documentation	2193
5.390std::basic_string< _CharT, _Traits, _Alloc > Class Template Reference	2196
5.390.1 Detailed Description	2201
5.390.2 Constructor & Destructor Documentation	2202
5.390.3 Member Function Documentation	2206

5.390.4 Member Data Documentation	2259
5.391std::basic_stringbuf< _CharT, _Traits, _Alloc > Class Template Reference	2259
5.391.1 Detailed Description	2263
5.391.2 Member Typedef Documentation	2263
5.391.3 Constructor & Destructor Documentation	2265
5.391.4 Member Function Documentation	2265
5.391.5 Member Data Documentation	2286
5.392std::basic_stringstream< _CharT, _Traits, _Alloc > Class Template Reference	2289
5.392.1 Detailed Description	2297
5.392.2 Member Typedef Documentation	2298
5.392.3 Member Enumeration Documentation	2304
5.392.4 Constructor & Destructor Documentation	2304
5.392.5 Member Function Documentation	2306
5.392.6 Member Data Documentation	2354
5.393std::bernoulli_distribution Class Reference	2362
5.393.1 Detailed Description	2363
5.393.2 Member Typedef Documentation	2363
5.393.3 Constructor & Destructor Documentation	2363
5.393.4 Member Function Documentation	2364
5.394std::bernoulli_distribution::param_type Struct Reference	2365
5.394.1 Detailed Description	2366
5.395std::bidirectional_iterator_tag Struct Reference	2366
5.395.1 Detailed Description	2367
5.396std::binary_function< _Arg1, _Arg2, _Result > Struct Template Reference	2367
5.396.1 Detailed Description	2368
5.396.2 Member Typedef Documentation	2368
5.397std::binary_negate< _Predicate > Class Template Reference	2369
5.397.1 Detailed Description	2369
5.397.2 Member Typedef Documentation	2369

5.398std::binder1st< _Operation > Class Template Reference	2370
5.398.1 Detailed Description	2371
5.398.2 Member Typedef Documentation	2372
5.399std::binder2nd< _Operation > Class Template Reference	2372
5.399.1 Detailed Description	2373
5.399.2 Member Typedef Documentation	2373
5.400std::binomial_distribution< _IntType > Class Template Reference . .	2374
5.400.1 Detailed Description	2375
5.400.2 Member Typedef Documentation	2375
5.400.3 Member Function Documentation	2375
5.400.4 Friends And Related Function Documentation	2378
5.401std::binomial_distribution< _IntType >::param_type Struct Reference	2379
5.401.1 Detailed Description	2379
5.402std::bitset< _Nb > Class Template Reference	2380
5.402.1 Detailed Description	2383
5.402.2 Constructor & Destructor Documentation	2384
5.402.3 Member Function Documentation	2386
5.403std::bitset< _Nb >::reference Class Reference	2394
5.403.1 Detailed Description	2394
5.404std::cauchy_distribution< _RealType > Class Template Reference . .	2395
5.404.1 Detailed Description	2395
5.404.2 Member Typedef Documentation	2396
5.404.3 Member Function Documentation	2396
5.405std::cauchy_distribution< _RealType >::param_type Struct Reference	2397
5.405.1 Detailed Description	2398
5.406std::char_traits< _CharT > Struct Template Reference	2398
5.406.1 Detailed Description	2400
5.407std::char_traits< __gnu_cxx::character< V, I, S > > Struct Template Reference	2400
5.407.1 Detailed Description	2401
5.408std::char_traits< char > Struct Template Reference	2401

5.408.1 Detailed Description	2402
5.409std::char_traits< wchar_t > Struct Template Reference	2402
5.409.1 Detailed Description	2403
5.410std::chi_squared_distribution< _RealType > Class Template Reference	2403
5.410.1 Detailed Description	2404
5.410.2 Member Typedef Documentation	2404
5.410.3 Member Function Documentation	2405
5.410.4 Friends And Related Function Documentation	2406
5.411std::chi_squared_distribution< _RealType >::param_type Struct Ref- erence	2407
5.411.1 Detailed Description	2408
5.412std::chrono::duration< _Rep, _Period > Struct Template Reference .	2408
5.412.1 Detailed Description	2409
5.413std::chrono::duration_values< _Rep > Struct Template Reference . .	2409
5.413.1 Detailed Description	2410
5.414std::chrono::system_clock Struct Reference	2410
5.414.1 Detailed Description	2411
5.415std::chrono::time_point< _Clock, _Dur > Struct Template Reference	2411
5.415.1 Detailed Description	2412
5.416std::chrono::treat_as_floating_point< _Rep > Struct Template Reference	2412
5.416.1 Detailed Description	2412
5.417std::codecvt< _InternT, _ExternT, _StateT > Class Template Reference	2413
5.417.1 Detailed Description	2415
5.417.2 Member Function Documentation	2415
5.418std::codecvt< _InternT, _ExternT, encoding_state > Class Template Reference	2418
5.418.1 Detailed Description	2420
5.418.2 Member Function Documentation	2421
5.419std::codecvt< char, char, mbstate_t > Class Template Reference . . .	2424
5.419.1 Detailed Description	2426
5.419.2 Member Function Documentation	2426
5.420std::codecvt< wchar_t, char, mbstate_t > Class Template Reference .	2429

5.420.1 Detailed Description	2431
5.420.2 Member Function Documentation	2431
5.421 <code>std::codecvt_base</code> Class Reference	2434
5.421.1 Detailed Description	2435
5.422 <code>std::codecvt_byname< _InternT, _ExternT, _StateT ></code> Class Template Reference	2435
5.422.1 Detailed Description	2438
5.422.2 Member Function Documentation	2438
5.423 <code>std::collate< _CharT ></code> Class Template Reference	2441
5.423.1 Detailed Description	2443
5.423.2 Member Typedef Documentation	2444
5.423.3 Constructor & Destructor Documentation	2444
5.423.4 Member Function Documentation	2445
5.423.5 Member Data Documentation	2448
5.424 <code>std::collate_byname< _CharT ></code> Class Template Reference	2449
5.424.1 Detailed Description	2450
5.424.2 Member Typedef Documentation	2451
5.424.3 Member Function Documentation	2451
5.424.4 Member Data Documentation	2454
5.425 <code>std::complex< _Tp ></code> Struct Template Reference	2455
5.425.1 Detailed Description	2455
5.425.2 Member Typedef Documentation	2456
5.425.3 Constructor & Destructor Documentation	2456
5.425.4 Member Function Documentation	2456
5.426 <code>std::condition_variable</code> Class Reference	2457
5.426.1 Detailed Description	2458
5.427 <code>std::condition_variable_any</code> Class Reference	2458
5.427.1 Detailed Description	2459
5.428 <code>std::conditional< _Cond, _Iftrue, _Iffalse ></code> Struct Template Reference	2459
5.428.1 Detailed Description	2459
5.429 <code>std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg ></code> Class Template Ref- erence	2459

5.429.1 Detailed Description	2460
5.429.2 Member Typedef Documentation	2461
5.430std::const_mem_fun1_t< _Ret, _Tp, _Arg > Class Template Reference	2461
5.430.1 Detailed Description	2462
5.430.2 Member Typedef Documentation	2463
5.431std::const_mem_fun_ref_t< _Ret, _Tp > Class Template Reference .	2463
5.431.1 Detailed Description	2464
5.431.2 Member Typedef Documentation	2465
5.432std::const_mem_fun_t< _Ret, _Tp > Class Template Reference . . .	2465
5.432.1 Detailed Description	2466
5.432.2 Member Typedef Documentation	2467
5.433std::ctype< _CharT > Class Template Reference	2467
5.433.1 Detailed Description	2470
5.433.2 Member Typedef Documentation	2470
5.433.3 Member Function Documentation	2471
5.433.4 Member Data Documentation	2483
5.434std::ctype< char > Class Template Reference	2484
5.434.1 Detailed Description	2486
5.434.2 Member Typedef Documentation	2487
5.434.3 Constructor & Destructor Documentation	2487
5.434.4 Member Function Documentation	2488
5.434.5 Member Data Documentation	2498
5.435std::ctype< wchar_t > Class Template Reference	2499
5.435.1 Detailed Description	2502
5.435.2 Member Typedef Documentation	2503
5.435.3 Constructor & Destructor Documentation	2503
5.435.4 Member Function Documentation	2504
5.435.5 Member Data Documentation	2516
5.436std::ctype_base Struct Reference	2516
5.436.1 Detailed Description	2517
5.437std::ctype_byname< _CharT > Class Template Reference	2518

5.437.1 Detailed Description	2520
5.437.2 Member Typedef Documentation	2520
5.437.3 Member Function Documentation	2521
5.437.4 Member Data Documentation	2533
5.438std::ctype_byname< char > Class Template Reference	2534
5.438.1 Detailed Description	2536
5.438.2 Member Typedef Documentation	2536
5.438.3 Member Function Documentation	2537
5.438.4 Member Data Documentation	2547
5.439std::decay< _Tp > Class Template Reference	2548
5.439.1 Detailed Description	2548
5.440std::decimal::decimal128 Class Reference	2548
5.440.1 Detailed Description	2550
5.440.2 Constructor & Destructor Documentation	2550
5.441std::decimal::decimal32 Class Reference	2550
5.441.1 Detailed Description	2552
5.441.2 Constructor & Destructor Documentation	2552
5.442std::decimal::decimal64 Class Reference	2552
5.442.1 Detailed Description	2554
5.442.2 Constructor & Destructor Documentation	2554
5.443std::default_delete< _Tp > Struct Template Reference	2554
5.443.1 Detailed Description	2554
5.444std::default_delete< _Tp[]> Struct Template Reference	2555
5.444.1 Detailed Description	2555
5.445std::defer_lock_t Struct Reference	2555
5.445.1 Detailed Description	2555
5.446std::deque< _Tp, _Alloc > Class Template Reference	2555
5.446.1 Detailed Description	2560
5.446.2 Constructor & Destructor Documentation	2562
5.446.3 Member Function Documentation	2565

5.447	<code>std::discard_block_engine< _RandomNumberEngine, __p, __r ></code>	
	Class Template Reference	2587
5.447.1	Detailed Description	2588
5.447.2	Member Typedef Documentation	2588
5.447.3	Constructor & Destructor Documentation	2588
5.447.4	Member Function Documentation	2590
5.447.5	Friends And Related Function Documentation	2592
5.448	<code>std::discrete_distribution< _IntType ></code> Class Template Reference . .	2594
5.448.1	Detailed Description	2595
5.448.2	Member Typedef Documentation	2595
5.448.3	Member Function Documentation	2595
5.448.4	Friends And Related Function Documentation	2597
5.449	<code>std::discrete_distribution< _IntType >::param_type</code> Struct Reference	2598
5.449.1	Detailed Description	2598
5.450	<code>std::divides< _Tp ></code> Struct Template Reference	2599
5.450.1	Detailed Description	2600
5.450.2	Member Typedef Documentation	2600
5.451	<code>std::domain_error</code> Class Reference	2601
5.451.1	Detailed Description	2601
5.451.2	Member Function Documentation	2601
5.452	<code>std::enable_if< bool, _Tp ></code> Struct Template Reference	2602
5.452.1	Detailed Description	2602
5.453	<code>std::enable_shared_from_this< _Tp ></code> Class Template Reference . . .	2602
5.453.1	Detailed Description	2603
5.454	<code>std::equal_to< _Tp ></code> Struct Template Reference	2603
5.454.1	Detailed Description	2604
5.454.2	Member Typedef Documentation	2605
5.455	<code>std::error_category</code> Class Reference	2605
5.455.1	Detailed Description	2606
5.456	<code>std::error_code</code> Struct Reference	2606
5.456.1	Detailed Description	2607

5.457	<code>std::error_condition</code> Struct Reference	2607
5.457.1	Detailed Description	2607
5.458	<code>std::exception</code> Class Reference	2607
5.458.1	Detailed Description	2609
5.458.2	Member Function Documentation	2609
5.459	<code>std::exponential_distribution<_RealType></code> Class Template Reference	2609
5.459.1	Detailed Description	2610
5.459.2	Member Typedef Documentation	2610
5.459.3	Constructor & Destructor Documentation	2611
5.459.4	Member Function Documentation	2611
5.460	<code>std::exponential_distribution<_RealType>::param_type</code> Struct Reference	2613
5.460.1	Detailed Description	2613
5.461	<code>std::extent<typename, _UInt></code> Struct Template Reference	2613
5.461.1	Detailed Description	2614
5.462	<code>std::extreme_value_distribution<_RealType></code> Class Template Reference	2615
5.462.1	Detailed Description	2616
5.462.2	Member Typedef Documentation	2616
5.462.3	Member Function Documentation	2616
5.463	<code>std::extreme_value_distribution<_RealType>::param_type</code> Struct Reference	2618
5.463.1	Detailed Description	2619
5.464	<code>std::fisher_f_distribution<_RealType></code> Class Template Reference	2619
5.464.1	Detailed Description	2620
5.464.2	Member Typedef Documentation	2620
5.464.3	Member Function Documentation	2620
5.464.4	Friends And Related Function Documentation	2622
5.465	<code>std::fisher_f_distribution<_RealType>::param_type</code> Struct Reference	2623
5.465.1	Detailed Description	2624
5.466	<code>std::forward_iterator_tag</code> Struct Reference	2624
5.466.1	Detailed Description	2625

5.467std::forward_list< _Tp, _Alloc > Class Template Reference	2625
5.467.1 Detailed Description	2628
5.467.2 Constructor & Destructor Documentation	2629
5.467.3 Member Function Documentation	2632
5.468std::fpos< _StateT > Class Template Reference	2649
5.468.1 Detailed Description	2650
5.468.2 Constructor & Destructor Documentation	2650
5.468.3 Member Function Documentation	2650
5.469std::front_insert_iterator< _Container > Class Template Reference	2652
5.469.1 Detailed Description	2653
5.469.2 Member Typedef Documentation	2653
5.469.3 Constructor & Destructor Documentation	2655
5.469.4 Member Function Documentation	2655
5.470std::function< _Res(_ArgTypes...)> Class Template Reference	2656
5.470.1 Detailed Description	2658
5.470.2 Constructor & Destructor Documentation	2658
5.470.3 Member Function Documentation	2660
5.471std::future< _Res > Class Template Reference	2665
5.471.1 Detailed Description	2666
5.471.2 Constructor & Destructor Documentation	2666
5.471.3 Member Function Documentation	2667
5.472std::future< _Res & > Class Template Reference	2667
5.472.1 Detailed Description	2669
5.472.2 Constructor & Destructor Documentation	2669
5.472.3 Member Function Documentation	2670
5.473std::future< void > Class Template Reference	2670
5.473.1 Detailed Description	2672
5.473.2 Constructor & Destructor Documentation	2672
5.473.3 Member Function Documentation	2673
5.474std::future_error Class Reference	2673
5.474.1 Detailed Description	2674

5.474.2 Member Function Documentation	2674
5.475std::gamma_distribution<_RealType> Class Template Reference . .	2675
5.475.1 Detailed Description	2676
5.475.2 Member Typedef Documentation	2676
5.475.3 Constructor & Destructor Documentation	2677
5.475.4 Member Function Documentation	2677
5.475.5 Friends And Related Function Documentation	2679
5.476std::gamma_distribution<_RealType>::param_type Struct Reference	2680
5.476.1 Detailed Description	2681
5.477std::geometric_distribution<_IntType> Class Template Reference .	2681
5.477.1 Detailed Description	2682
5.477.2 Member Typedef Documentation	2682
5.477.3 Member Function Documentation	2682
5.478std::geometric_distribution<_IntType>::param_type Struct Reference	2684
5.478.1 Detailed Description	2685
5.479std::greater<_Tp> Struct Template Reference	2685
5.479.1 Detailed Description	2686
5.479.2 Member Typedef Documentation	2686
5.480std::greater_equal<_Tp> Struct Template Reference	2687
5.480.1 Detailed Description	2687
5.480.2 Member Typedef Documentation	2688
5.481std::gslice Class Reference	2688
5.481.1 Detailed Description	2689
5.482std::gslice_array<_Tp> Class Template Reference	2689
5.482.1 Detailed Description	2690
5.483std::has_nothrow_copy_assign<_Tp> Struct Template Reference . .	2691
5.483.1 Detailed Description	2692
5.484std::has_nothrow_copy_constructor<_Tp> Struct Template Reference	2692
5.484.1 Detailed Description	2693
5.485std::has_nothrow_default_constructor<_Tp> Struct Template Refer- ence	2694

5.485.1 Detailed Description	2695
5.486std::has_trivial_copy_assign< _Tp > Struct Template Reference . . .	2695
5.486.1 Detailed Description	2696
5.487std::has_trivial_copy_constructor< _Tp > Struct Template Reference	2696
5.487.1 Detailed Description	2697
5.488std::has_trivial_default_constructor< _Tp > Struct Template Reference	2698
5.488.1 Detailed Description	2699
5.489std::has_trivial_destructor< _Tp > Struct Template Reference	2699
5.489.1 Detailed Description	2700
5.490std::has_virtual_destructor< _Tp > Struct Template Reference	2700
5.490.1 Detailed Description	2701
5.491std::hash< _Tp > Struct Template Reference	2702
5.491.1 Detailed Description	2703
5.492std::hash< __debug::bitset< _Nb > > Struct Template Reference . .	2703
5.492.1 Detailed Description	2703
5.493std::hash< __debug::vector< bool, _Alloc > > Struct Template Ref- erence	2704
5.493.1 Detailed Description	2704
5.494std::hash< __gnu_cxx::throw_value_limit > Struct Template Reference	2704
5.494.1 Detailed Description	2705
5.494.2 Member Typedef Documentation	2706
5.495std::hash< __gnu_cxx::throw_value_random > Struct Template Ref- erence	2706
5.495.1 Detailed Description	2707
5.495.2 Member Typedef Documentation	2708
5.496std::hash< __profile::bitset< _Nb > > Struct Template Reference . .	2708
5.496.1 Detailed Description	2709
5.497std::hash< __profile::vector< bool, _Alloc > > Struct Template Ref- erence	2709
5.497.1 Detailed Description	2709
5.498std::hash< __shared_ptr< _Tp, _Lp > > Struct Template Reference .	2709
5.498.1 Detailed Description	2710

5.498.2 Member Typedef Documentation	2711
5.499std::hash< _Tp * > Struct Template Reference	2711
5.499.1 Detailed Description	2711
5.500std::hash< error_code > Struct Template Reference	2712
5.500.1 Detailed Description	2712
5.501std::hash< shared_ptr< _Tp > > Struct Template Reference	2712
5.501.1 Detailed Description	2713
5.501.2 Member Typedef Documentation	2714
5.502std::hash< string > Struct Template Reference	2714
5.502.1 Detailed Description	2714
5.503std::hash< thread::id > Struct Template Reference	2715
5.503.1 Detailed Description	2715
5.504std::hash< type_index > Struct Template Reference	2715
5.504.1 Detailed Description	2716
5.505std::hash< u16string > Struct Template Reference	2716
5.505.1 Detailed Description	2716
5.506std::hash< u32string > Struct Template Reference	2716
5.506.1 Detailed Description	2717
5.507std::hash< unique_ptr< _Tp, _Dp > > Struct Template Reference	2717
5.507.1 Detailed Description	2718
5.507.2 Member Typedef Documentation	2719
5.508std::hash< wstring > Struct Template Reference	2719
5.508.1 Detailed Description	2719
5.509std::hash<::bitset< _Nb > > Struct Template Reference	2720
5.509.1 Detailed Description	2720
5.510std::hash<::vector< bool, _Alloc > > Struct Template Reference	2720
5.510.1 Detailed Description	2721
5.511std::independent_bits_engine< _RandomNumberEngine, __w, _- UIntType > Class Template Reference	2721
5.511.1 Detailed Description	2722
5.511.2 Member Typedef Documentation	2722

5.511.3 Constructor & Destructor Documentation	2722
5.511.4 Member Function Documentation	2724
5.511.5 Friends And Related Function Documentation	2726
5.512std::indirect_array< _Tp > Class Template Reference	2727
5.512.1 Detailed Description	2729
5.512.2 Member Function Documentation	2729
5.513std::initializer_list< _E > Class Template Reference	2731
5.513.1 Detailed Description	2731
5.514std::input_iterator_tag Struct Reference	2732
5.514.1 Detailed Description	2732
5.515std::insert_iterator< _Container > Class Template Reference	2732
5.515.1 Detailed Description	2734
5.515.2 Member Typedef Documentation	2734
5.515.3 Constructor & Destructor Documentation	2735
5.515.4 Member Function Documentation	2735
5.516std::integral_constant< _Tp, __v > Struct Template Reference	2737
5.516.1 Detailed Description	2738
5.517std::invalid_argument Class Reference	2738
5.517.1 Detailed Description	2738
5.517.2 Member Function Documentation	2739
5.518std::ios_base Class Reference	2739
5.518.1 Detailed Description	2742
5.518.2 Member Typedef Documentation	2742
5.518.3 Member Enumeration Documentation	2745
5.518.4 Constructor & Destructor Documentation	2745
5.518.5 Member Function Documentation	2745
5.518.6 Member Data Documentation	2752
5.519std::ios_base::failure Class Reference	2760
5.519.1 Detailed Description	2761
5.519.2 Member Function Documentation	2761
5.520std::is_abstract< _Tp > Struct Template Reference	2761

5.520.1 Detailed Description	2762
5.521std::is_arithmetic< _Tp > Struct Template Reference	2763
5.521.1 Detailed Description	2763
5.522std::is_array< typename > Struct Template Reference	2764
5.522.1 Detailed Description	2764
5.523std::is_base_of< _Base, _Derived > Struct Template Reference	2765
5.523.1 Detailed Description	2766
5.524std::is_bind_expression< _Tp > Struct Template Reference	2766
5.524.1 Detailed Description	2767
5.525std::is_bind_expression< _Bind< _Signature > > Struct Template Reference	2767
5.525.1 Detailed Description	2768
5.526std::is_bind_expression< _Bind_result< _Result, _Signature > > Struct Template Reference	2768
5.526.1 Detailed Description	2769
5.527std::is_class< _Tp > Struct Template Reference	2769
5.527.1 Detailed Description	2770
5.528std::is_compound< _Tp > Struct Template Reference	2771
5.528.1 Detailed Description	2772
5.529std::is_const< typename > Struct Template Reference	2772
5.529.1 Detailed Description	2773
5.530std::is_constructible< _Tp, _Args > Struct Template Reference	2773
5.530.1 Detailed Description	2773
5.531std::is_convertible< _From, _To > Struct Template Reference	2774
5.531.1 Detailed Description	2775
5.532std::is_empty< _Tp > Struct Template Reference	2775
5.532.1 Detailed Description	2776
5.533std::is_enum< _Tp > Struct Template Reference	2776
5.533.1 Detailed Description	2777
5.534std::is_error_code_enum< _Tp > Struct Template Reference	2778
5.534.1 Detailed Description	2779
5.535std::is_error_code_enum< future_errc > Struct Template Reference	2779

5.535.1 Detailed Description	2780
5.536std::is_error_condition_enum< _Tp > Struct Template Reference	2780
5.536.1 Detailed Description	2781
5.537std::is_explicitly_convertible< _From, _To > Struct Template Reference	2781
5.537.1 Detailed Description	2782
5.538std::is_floating_point< _Tp > Struct Template Reference	2782
5.538.1 Detailed Description	2782
5.539std::is_function< typename > Struct Template Reference	2783
5.539.1 Detailed Description	2783
5.540std::is_fundamental< _Tp > Struct Template Reference	2784
5.540.1 Detailed Description	2784
5.541std::is_integral< _Tp > Struct Template Reference	2785
5.541.1 Detailed Description	2785
5.542std::is_literal_type< _Tp > Struct Template Reference	2785
5.542.1 Detailed Description	2786
5.543std::is_lvalue_reference< typename > Struct Template Reference	2787
5.543.1 Detailed Description	2788
5.544std::is_member_function_pointer< _Tp > Struct Template Reference	2788
5.544.1 Detailed Description	2788
5.545std::is_member_object_pointer< _Tp > Struct Template Reference	2789
5.545.1 Detailed Description	2789
5.546std::is_nothrow_constructible< _Tp, _Args > Struct Template Reference	2789
5.546.1 Detailed Description	2790
5.547std::is_object< _Tp > Struct Template Reference	2790
5.547.1 Detailed Description	2791
5.548std::is_placeholder< _Tp > Struct Template Reference	2791
5.548.1 Detailed Description	2792
5.549std::is_placeholder< _Placeholder< _Num > > Struct Template Reference	2793
5.549.1 Detailed Description	2794
5.550std::is_pod< _Tp > Struct Template Reference	2794

5.550.1 Detailed Description	2795
5.551std::is_pointer< _Tp > Struct Template Reference	2795
5.551.1 Detailed Description	2796
5.552std::is_polymorphic< _Tp > Struct Template Reference	2796
5.552.1 Detailed Description	2797
5.553std::is_reference< _Tp > Struct Template Reference	2797
5.553.1 Detailed Description	2798
5.554std::is_rvalue_reference< typename > Struct Template Reference . .	2798
5.554.1 Detailed Description	2799
5.555std::is_same< typename, typename > Struct Template Reference . . .	2799
5.555.1 Detailed Description	2800
5.556std::is_scalar< _Tp > Struct Template Reference	2800
5.556.1 Detailed Description	2801
5.557std::is_signed< _Tp > Struct Template Reference	2801
5.557.1 Detailed Description	2802
5.558std::is_standard_layout< _Tp > Struct Template Reference	2802
5.558.1 Detailed Description	2803
5.559std::is_trivial< _Tp > Struct Template Reference	2803
5.559.1 Detailed Description	2804
5.560std::is_union< _Tp > Struct Template Reference	2805
5.560.1 Detailed Description	2806
5.561std::is_unsigned< _Tp > Struct Template Reference	2806
5.561.1 Detailed Description	2806
5.562std::is_void< _Tp > Struct Template Reference	2807
5.562.1 Detailed Description	2807
5.563std::is_volatile< typename > Struct Template Reference	2807
5.563.1 Detailed Description	2808
5.564std::istream_iterator< _Tp, _CharT, _Traits, _Dist > Class Template Reference	2809
5.564.1 Detailed Description	2810
5.564.2 Member Typedef Documentation	2810

5.564.3 Constructor & Destructor Documentation	2811
5.565std::istreambuf_iterator< _CharT, _Traits > Class Template Reference	2811
5.565.1 Detailed Description	2813
5.565.2 Member Typedef Documentation	2813
5.565.3 Constructor & Destructor Documentation	2815
5.565.4 Member Function Documentation	2816
5.566std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference > Struct Template Reference	2817
5.566.1 Detailed Description	2817
5.566.2 Member Typedef Documentation	2818
5.567std::iterator_traits< _Tp * > Struct Template Reference	2819
5.567.1 Detailed Description	2819
5.568std::iterator_traits< const _Tp * > Struct Template Reference	2819
5.568.1 Detailed Description	2820
5.569std::length_error Class Reference	2820
5.569.1 Detailed Description	2821
5.569.2 Member Function Documentation	2821
5.570std::less< _Tp > Struct Template Reference	2821
5.570.1 Detailed Description	2822
5.570.2 Member Typedef Documentation	2823
5.571std::less_equal< _Tp > Struct Template Reference	2823
5.571.1 Detailed Description	2824
5.571.2 Member Typedef Documentation	2825
5.572std::linear_congruential_engine< _UIntType, __a, __c, __m > Class Template Reference	2825
5.572.1 Detailed Description	2827
5.572.2 Member Typedef Documentation	2827
5.572.3 Constructor & Destructor Documentation	2827
5.572.4 Member Function Documentation	2828
5.572.5 Friends And Related Function Documentation	2830
5.572.6 Member Data Documentation	2832
5.573std::list< _Tp, _Alloc > Class Template Reference	2832

5.573.1 Detailed Description	2836
5.573.2 Constructor & Destructor Documentation	2837
5.573.3 Member Function Documentation	2840
5.574std::locale Class Reference	2860
5.574.1 Detailed Description	2861
5.574.2 Member Typedef Documentation	2862
5.574.3 Constructor & Destructor Documentation	2862
5.574.4 Member Function Documentation	2864
5.574.5 Friends And Related Function Documentation	2867
5.574.6 Member Data Documentation	2868
5.575std::locale::facet Class Reference	2870
5.575.1 Detailed Description	2872
5.575.2 Constructor & Destructor Documentation	2872
5.576std::locale::id Class Reference	2872
5.576.1 Detailed Description	2873
5.576.2 Constructor & Destructor Documentation	2873
5.576.3 Friends And Related Function Documentation	2873
5.577std::lock_guard< _Mutex > Class Template Reference	2874
5.577.1 Detailed Description	2875
5.578std::logic_error Class Reference	2875
5.578.1 Detailed Description	2876
5.578.2 Constructor & Destructor Documentation	2876
5.578.3 Member Function Documentation	2876
5.579std::logical_and< _Tp > Struct Template Reference	2876
5.579.1 Detailed Description	2877
5.579.2 Member Typedef Documentation	2878
5.580std::logical_not< _Tp > Struct Template Reference	2878
5.580.1 Detailed Description	2879
5.580.2 Member Typedef Documentation	2880
5.581std::logical_or< _Tp > Struct Template Reference	2880
5.581.1 Detailed Description	2881

5.581.2 Member Typedef Documentation	2882
5.582std::lognormal_distribution< _RealType > Class Template Reference	2882
5.582.1 Detailed Description	2883
5.582.2 Member Typedef Documentation	2884
5.582.3 Member Function Documentation	2884
5.582.4 Friends And Related Function Documentation	2885
5.583std::lognormal_distribution< _RealType >::param_type Struct Reference	2887
5.583.1 Detailed Description	2887
5.584std::make_signed< _Tp > Struct Template Reference	2887
5.584.1 Detailed Description	2887
5.585std::make_unsigned< _Tp > Struct Template Reference	2888
5.585.1 Detailed Description	2888
5.586std::map< _Key, _Tp, _Compare, _Alloc > Class Template Reference	2888
5.586.1 Detailed Description	2891
5.586.2 Constructor & Destructor Documentation	2891
5.586.3 Member Function Documentation	2894
5.587std::mask_array< _Tp > Class Template Reference	2910
5.587.1 Detailed Description	2911
5.587.2 Member Function Documentation	2911
5.588std::match_results< _Bi_iter, _Allocator > Class Template Reference	2913
5.588.1 Detailed Description	2917
5.588.2 Constructor & Destructor Documentation	2918
5.588.3 Member Function Documentation	2918
5.589std::mem_fun1_ref_t< _Ret, _Tp, _Arg > Class Template Reference	2925
5.589.1 Detailed Description	2926
5.589.2 Member Typedef Documentation	2926
5.590std::mem_fun1_t< _Ret, _Tp, _Arg > Class Template Reference	2926
5.590.1 Detailed Description	2927
5.590.2 Member Typedef Documentation	2928
5.591std::mem_fun_ref_t< _Ret, _Tp > Class Template Reference	2928

5.591.1 Detailed Description	2929
5.591.2 Member Typedef Documentation	2930
5.592std::mem_fun_t< _Ret, _Tp > Class Template Reference	2930
5.592.1 Detailed Description	2931
5.592.2 Member Typedef Documentation	2932
5.593std::messages< _CharT > Class Template Reference	2932
5.593.1 Detailed Description	2934
5.593.2 Member Typedef Documentation	2935
5.593.3 Constructor & Destructor Documentation	2935
5.593.4 Member Data Documentation	2936
5.594std::messages_base Struct Reference	2936
5.594.1 Detailed Description	2937
5.595std::messages_byname< _CharT > Class Template Reference	2937
5.595.1 Detailed Description	2939
5.595.2 Member Typedef Documentation	2940
5.595.3 Member Data Documentation	2940
5.596std::minus< _Tp > Struct Template Reference	2940
5.596.1 Detailed Description	2941
5.596.2 Member Typedef Documentation	2942
5.597std::modulus< _Tp > Struct Template Reference	2942
5.597.1 Detailed Description	2943
5.597.2 Member Typedef Documentation	2944
5.598std::money_base Class Reference	2944
5.598.1 Detailed Description	2945
5.599std::money_get< _CharT, _InIter > Class Template Reference	2946
5.599.1 Detailed Description	2947
5.599.2 Member Typedef Documentation	2948
5.599.3 Constructor & Destructor Documentation	2948
5.599.4 Member Function Documentation	2949
5.599.5 Member Data Documentation	2951
5.600std::money_put< _CharT, _OutIter > Class Template Reference	2952

5.600.1 Detailed Description	2953
5.600.2 Member Typedef Documentation	2954
5.600.3 Constructor & Destructor Documentation	2954
5.600.4 Member Function Documentation	2955
5.600.5 Member Data Documentation	2958
5.601std::moneypunct< _CharT, _Intl > Class Template Reference	2958
5.601.1 Detailed Description	2960
5.601.2 Member Typedef Documentation	2961
5.601.3 Constructor & Destructor Documentation	2961
5.601.4 Member Function Documentation	2963
5.601.5 Member Data Documentation	2971
5.602std::moneypunct_byname< _CharT, _Intl > Class Template Reference	2972
5.602.1 Detailed Description	2974
5.602.2 Member Typedef Documentation	2974
5.602.3 Member Function Documentation	2975
5.602.4 Member Data Documentation	2983
5.603std::move_iterator< _Iterator > Class Template Reference	2984
5.603.1 Detailed Description	2985
5.604std::multimap< _Key, _Tp, _Compare, _Alloc > Class Template Reference	2985
5.604.1 Detailed Description	2987
5.604.2 Constructor & Destructor Documentation	2988
5.604.3 Member Function Documentation	2990
5.605std::multiplies< _Tp > Struct Template Reference	3004
5.605.1 Detailed Description	3005
5.605.2 Member Typedef Documentation	3006
5.606std::multiset< _Key, _Compare, _Alloc > Class Template Reference	3006
5.606.1 Detailed Description	3008
5.606.2 Constructor & Destructor Documentation	3009
5.606.3 Member Function Documentation	3011
5.606.4 Friends And Related Function Documentation	3024

5.607std::mutex Class Reference	3025
5.607.1 Detailed Description	3026
5.608std::negate< _Tp > Struct Template Reference	3026
5.608.1 Detailed Description	3027
5.608.2 Member Typedef Documentation	3027
5.609std::negative_binomial_distribution< _IntType > Class Template Reference	3027
5.609.1 Detailed Description	3029
5.609.2 Member Typedef Documentation	3029
5.609.3 Member Function Documentation	3029
5.609.4 Friends And Related Function Documentation	3031
5.610std::negative_binomial_distribution< _IntType >::param_type Struct Reference	3032
5.610.1 Detailed Description	3033
5.611std::nested_exception Class Reference	3033
5.611.1 Detailed Description	3033
5.612std::normal_distribution< _RealType > Class Template Reference	3034
5.612.1 Detailed Description	3035
5.612.2 Member Typedef Documentation	3035
5.612.3 Constructor & Destructor Documentation	3036
5.612.4 Member Function Documentation	3036
5.612.5 Friends And Related Function Documentation	3038
5.613std::normal_distribution< _RealType >::param_type Struct Reference	3039
5.613.1 Detailed Description	3040
5.614std::not_equal_to< _Tp > Struct Template Reference	3040
5.614.1 Detailed Description	3041
5.614.2 Member Typedef Documentation	3042
5.615std::num_get< _CharT, _InIter > Class Template Reference	3042
5.615.1 Detailed Description	3045
5.615.2 Member Typedef Documentation	3045
5.615.3 Constructor & Destructor Documentation	3046
5.615.4 Member Function Documentation	3046

5.615.5 Member Data Documentation	3063
5.616std::num_put< _CharT, _OutIter > Class Template Reference	3063
5.616.1 Detailed Description	3065
5.616.2 Member Typedef Documentation	3066
5.616.3 Constructor & Destructor Documentation	3066
5.616.4 Member Function Documentation	3067
5.616.5 Member Data Documentation	3078
5.617std::numeric_limits< _Tp > Struct Template Reference	3078
5.617.1 Detailed Description	3080
5.617.2 Member Function Documentation	3080
5.617.3 Member Data Documentation	3082
5.618std::numeric_limits< bool > Struct Template Reference	3086
5.618.1 Detailed Description	3087
5.619std::numeric_limits< char > Struct Template Reference	3087
5.619.1 Detailed Description	3089
5.620std::numeric_limits< char16_t > Struct Template Reference	3089
5.620.1 Detailed Description	3090
5.621std::numeric_limits< char32_t > Struct Template Reference	3090
5.621.1 Detailed Description	3091
5.622std::numeric_limits< double > Struct Template Reference	3091
5.622.1 Detailed Description	3093
5.623std::numeric_limits< float > Struct Template Reference	3093
5.623.1 Detailed Description	3094
5.624std::numeric_limits< int > Struct Template Reference	3094
5.624.1 Detailed Description	3095
5.625std::numeric_limits< long > Struct Template Reference	3095
5.625.1 Detailed Description	3097
5.626std::numeric_limits< long double > Struct Template Reference	3097
5.626.1 Detailed Description	3098
5.627std::numeric_limits< long long > Struct Template Reference	3098
5.627.1 Detailed Description	3099

5.628std::numeric_limits< short > Struct Template Reference	3099
5.628.1 Detailed Description	3101
5.629std::numeric_limits< signed char > Struct Template Reference	3101
5.629.1 Detailed Description	3102
5.630std::numeric_limits< unsigned char > Struct Template Reference . . .	3102
5.630.1 Detailed Description	3103
5.631std::numeric_limits< unsigned int > Struct Template Reference . . .	3103
5.631.1 Detailed Description	3105
5.632std::numeric_limits< unsigned long > Struct Template Reference . . .	3105
5.632.1 Detailed Description	3106
5.633std::numeric_limits< unsigned long long > Struct Template Reference	3106
5.633.1 Detailed Description	3107
5.634std::numeric_limits< unsigned short > Struct Template Reference . .	3107
5.634.1 Detailed Description	3109
5.635std::numeric_limits< wchar_t > Struct Template Reference	3109
5.635.1 Detailed Description	3110
5.636std::num_punct< _CharT > Class Template Reference	3110
5.636.1 Detailed Description	3112
5.636.2 Member Typedef Documentation	3113
5.636.3 Constructor & Destructor Documentation	3113
5.636.4 Member Function Documentation	3114
5.636.5 Member Data Documentation	3119
5.637std::num_punct_byname< _CharT > Class Template Reference	3119
5.637.1 Detailed Description	3121
5.637.2 Member Typedef Documentation	3121
5.637.3 Member Function Documentation	3121
5.637.4 Member Data Documentation	3126
5.638std::once_flag Struct Reference	3126
5.638.1 Detailed Description	3126
5.638.2 Friends And Related Function Documentation	3126
5.639std::ostream_iterator< _Tp, _CharT, _Traits > Class Template Reference	3127

5.639.1 Detailed Description	3128
5.639.2 Member Typedef Documentation	3128
5.639.3 Constructor & Destructor Documentation	3130
5.639.4 Member Function Documentation	3131
5.640std::ostreambuf_iterator< _CharT, _Traits > Class Template Reference	3131
5.640.1 Detailed Description	3133
5.640.2 Member Typedef Documentation	3133
5.640.3 Constructor & Destructor Documentation	3135
5.640.4 Member Function Documentation	3135
5.641std::out_of_range Class Reference	3137
5.641.1 Detailed Description	3137
5.641.2 Member Function Documentation	3137
5.642std::output_iterator_tag Struct Reference	3138
5.642.1 Detailed Description	3138
5.643std::overflow_error Class Reference	3138
5.643.1 Detailed Description	3139
5.643.2 Member Function Documentation	3139
5.644std::owner_less< shared_ptr< _Tp > > Struct Template Reference .	3139
5.644.1 Detailed Description	3140
5.644.2 Member Typedef Documentation	3140
5.645std::owner_less< weak_ptr< _Tp > > Struct Template Reference . .	3140
5.645.1 Detailed Description	3141
5.645.2 Member Typedef Documentation	3141
5.646std::packaged_task< _Res(_ArgTypes...)> Class Template Reference	3142
5.646.1 Detailed Description	3142
5.647std::pair< _T1, _T2 > Struct Template Reference	3143
5.647.1 Detailed Description	3144
5.647.2 Member Typedef Documentation	3144
5.647.3 Constructor & Destructor Documentation	3144
5.647.4 Member Data Documentation	3145

5.648	<code>std::piecewise_constant_distribution< _RealType ></code> Class Template Reference	3146
5.648.1	Detailed Description	3147
5.648.2	Member Typedef Documentation	3147
5.648.3	Member Function Documentation	3147
5.648.4	Friends And Related Function Documentation	3149
5.649	<code>std::piecewise_constant_distribution< _RealType >::param_type</code> Struct Reference	3150
5.649.1	Detailed Description	3151
5.650	<code>std::piecewise_construct_t</code> Struct Reference	3151
5.650.1	Detailed Description	3151
5.651	<code>std::piecewise_linear_distribution< _RealType ></code> Class Template Reference	3152
5.651.1	Detailed Description	3153
5.651.2	Member Typedef Documentation	3153
5.651.3	Member Function Documentation	3153
5.651.4	Friends And Related Function Documentation	3156
5.652	<code>std::piecewise_linear_distribution< _RealType >::param_type</code> Struct Reference	3157
5.652.1	Detailed Description	3157
5.653	<code>std::plus< _Tp ></code> Struct Template Reference	3158
5.653.1	Detailed Description	3158
5.653.2	Member Typedef Documentation	3159
5.654	<code>std::pointer_to_binary_function< _Arg1, _Arg2, _Result ></code> Class Template Reference	3159
5.654.1	Detailed Description	3160
5.654.2	Member Typedef Documentation	3161
5.655	<code>std::pointer_to_unary_function< _Arg, _Result ></code> Class Template Reference	3161
5.655.1	Detailed Description	3162
5.655.2	Member Typedef Documentation	3163
5.656	<code>std::poisson_distribution< _IntType ></code> Class Template Reference	3163
5.656.1	Detailed Description	3164

5.656.2 Member Typedef Documentation	3164
5.656.3 Member Function Documentation	3165
5.656.4 Friends And Related Function Documentation	3167
5.657std::poisson_distribution< _IntType >::param_type Struct Reference	3168
5.657.1 Detailed Description	3168
5.658std::priority_queue< _Tp, _Sequence, _Compare > Class Template Reference	3169
5.658.1 Detailed Description	3170
5.658.2 Constructor & Destructor Documentation	3170
5.658.3 Member Function Documentation	3171
5.659std::promise< _Res > Class Template Reference	3173
5.659.1 Detailed Description	3174
5.660std::promise< _Res & > Class Template Reference	3174
5.660.1 Detailed Description	3174
5.661std::promise< void > Class Template Reference	3175
5.661.1 Detailed Description	3175
5.662std::queue< _Tp, _Sequence > Class Template Reference	3175
5.662.1 Detailed Description	3177
5.662.2 Constructor & Destructor Documentation	3177
5.662.3 Member Function Documentation	3177
5.662.4 Member Data Documentation	3179
5.663std::random_access_iterator_tag Struct Reference	3180
5.663.1 Detailed Description	3180
5.664std::random_device Class Reference	3181
5.664.1 Detailed Description	3181
5.664.2 Member Typedef Documentation	3181
5.665std::range_error Class Reference	3182
5.665.1 Detailed Description	3182
5.665.2 Member Function Documentation	3182
5.666std::rank< typename > Struct Template Reference	3183
5.666.1 Detailed Description	3184

5.667std::ratio< _Num, _Den > Struct Template Reference	3184
5.667.1 Detailed Description	3184
5.668std::ratio_add< _R1, _R2 > Struct Template Reference	3185
5.668.1 Detailed Description	3185
5.669std::ratio_divide< _R1, _R2 > Struct Template Reference	3185
5.669.1 Detailed Description	3186
5.670std::ratio_equal< _R1, _R2 > Struct Template Reference	3186
5.670.1 Detailed Description	3187
5.671std::ratio_multiply< _R1, _R2 > Struct Template Reference	3187
5.671.1 Detailed Description	3188
5.672std::ratio_not_equal< _R1, _R2 > Struct Template Reference	3188
5.672.1 Detailed Description	3189
5.673std::ratio_subtract< _R1, _R2 > Struct Template Reference	3189
5.673.1 Detailed Description	3189
5.674std::raw_storage_iterator< _OutputIterator, _Tp > Class Template Reference	3190
5.674.1 Detailed Description	3191
5.674.2 Member Typedef Documentation	3191
5.675std::recursive_mutex Class Reference	3192
5.675.1 Detailed Description	3192
5.676std::recursive_timed_mutex Class Reference	3193
5.676.1 Detailed Description	3193
5.677std::reference_wrapper< _Tp > Class Template Reference	3193
5.677.1 Detailed Description	3194
5.678std::regex_error Class Reference	3195
5.678.1 Detailed Description	3196
5.678.2 Constructor & Destructor Documentation	3196
5.678.3 Member Function Documentation	3196
5.679std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > Class Template Reference	3197
5.679.1 Detailed Description	3197
5.679.2 Constructor & Destructor Documentation	3198

5.679.3 Member Function Documentation	3199
5.680std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > Class Template Reference	3201
5.680.1 Detailed Description	3202
5.680.2 Constructor & Destructor Documentation	3202
5.680.3 Member Function Documentation	3205
5.681std::regex_traits< _Ch_type > Struct Template Reference	3207
5.681.1 Detailed Description	3208
5.681.2 Constructor & Destructor Documentation	3208
5.681.3 Member Function Documentation	3209
5.682std::remove_all_extents< _Tp > Struct Template Reference	3214
5.682.1 Detailed Description	3214
5.683std::remove_const< _Tp > Struct Template Reference	3214
5.683.1 Detailed Description	3214
5.684std::remove_cv< _Tp > Struct Template Reference	3215
5.684.1 Detailed Description	3215
5.685std::remove_extent< _Tp > Struct Template Reference	3215
5.685.1 Detailed Description	3215
5.686std::remove_pointer< _Tp > Struct Template Reference	3216
5.686.1 Detailed Description	3216
5.687std::remove_reference< _Tp > Struct Template Reference	3216
5.687.1 Detailed Description	3216
5.688std::remove_volatile< _Tp > Struct Template Reference	3216
5.688.1 Detailed Description	3217
5.689std::reverse_iterator< _Iterator > Class Template Reference	3217
5.689.1 Detailed Description	3218
5.689.2 Member Typedef Documentation	3219
5.689.3 Constructor & Destructor Documentation	3220
5.689.4 Member Function Documentation	3221
5.690std::runtime_error Class Reference	3225
5.690.1 Detailed Description	3226

5.690.2 Constructor & Destructor Documentation	3226
5.690.3 Member Function Documentation	3226
5.691std::seed_seq Class Reference	3226
5.691.1 Detailed Description	3227
5.691.2 Member Typedef Documentation	3227
5.691.3 Constructor & Destructor Documentation	3227
5.692std::set< _Key, _Compare, _Alloc > Class Template Reference . . .	3228
5.692.1 Detailed Description	3230
5.692.2 Member Typedef Documentation	3231
5.692.3 Constructor & Destructor Documentation	3234
5.692.4 Member Function Documentation	3237
5.693std::shared_future< _Res > Class Template Reference	3250
5.693.1 Detailed Description	3251
5.693.2 Constructor & Destructor Documentation	3251
5.693.3 Member Function Documentation	3252
5.694std::shared_future< _Res & > Class Template Reference	3253
5.694.1 Detailed Description	3254
5.694.2 Constructor & Destructor Documentation	3254
5.694.3 Member Function Documentation	3255
5.695std::shared_future< void > Class Template Reference	3255
5.695.1 Detailed Description	3257
5.695.2 Constructor & Destructor Documentation	3257
5.695.3 Member Function Documentation	3258
5.696std::shared_ptr< _Tp > Class Template Reference	3258
5.696.1 Detailed Description	3261
5.696.2 Constructor & Destructor Documentation	3261
5.696.3 Friends And Related Function Documentation	3266
5.697std::shuffle_order_engine< _RandomNumberEngine, __k > Class Template Reference	3267
5.697.1 Detailed Description	3268
5.697.2 Member Typedef Documentation	3268

5.697.3 Constructor & Destructor Documentation	3269
5.697.4 Member Function Documentation	3270
5.697.5 Friends And Related Function Documentation	3272
5.698std::slice Class Reference	3274
5.698.1 Detailed Description	3274
5.699std::slice_array< _Tp, _Sequence > Class Template Reference	3274
5.699.1 Detailed Description	3275
5.699.2 Member Function Documentation	3276
5.700std::stack< _Tp, _Sequence > Class Template Reference	3277
5.700.1 Detailed Description	3279
5.700.2 Constructor & Destructor Documentation	3279
5.700.3 Member Function Documentation	3280
5.701std::student_t_distribution< _RealType > Class Template Reference	3281
5.701.1 Detailed Description	3282
5.701.2 Member Typedef Documentation	3282
5.701.3 Member Function Documentation	3283
5.701.4 Friends And Related Function Documentation	3284
5.702std::student_t_distribution< _RealType >::param_type Struct Reference	3285
5.702.1 Detailed Description	3286
5.703std::sub_match< _BiIter > Class Template Reference	3287
5.703.1 Detailed Description	3288
5.703.2 Member Typedef Documentation	3288
5.703.3 Member Function Documentation	3288
5.703.4 Member Data Documentation	3291
5.704std::system_error Class Reference	3291
5.704.1 Detailed Description	3292
5.704.2 Member Function Documentation	3292
5.705std::thread Class Reference	3293
5.705.1 Detailed Description	3294
5.705.2 Member Function Documentation	3294
5.706std::thread::id Class Reference	3294

5.706.1 Detailed Description	3294
5.707std::time_base Class Reference	3295
5.707.1 Detailed Description	3295
5.708std::time_get< _CharT, _InIter > Class Template Reference	3296
5.708.1 Detailed Description	3298
5.708.2 Member Typedef Documentation	3298
5.708.3 Constructor & Destructor Documentation	3299
5.708.4 Member Function Documentation	3299
5.708.5 Member Data Documentation	3308
5.709std::time_get_byname< _CharT, _InIter > Class Template Reference	3308
5.709.1 Detailed Description	3310
5.709.2 Member Typedef Documentation	3310
5.709.3 Member Function Documentation	3311
5.709.4 Member Data Documentation	3319
5.710std::time_put< _CharT, _OutIter > Class Template Reference	3319
5.710.1 Detailed Description	3321
5.710.2 Member Typedef Documentation	3321
5.710.3 Constructor & Destructor Documentation	3322
5.710.4 Member Function Documentation	3322
5.710.5 Member Data Documentation	3324
5.711std::time_put_byname< _CharT, _OutIter > Class Template Reference	3325
5.711.1 Detailed Description	3326
5.711.2 Member Typedef Documentation	3326
5.711.3 Member Function Documentation	3327
5.711.4 Member Data Documentation	3329
5.712std::timed_mutex Class Reference	3329
5.712.1 Detailed Description	3330
5.713std::try_to_lock_t Struct Reference	3330
5.713.1 Detailed Description	3330
5.714std::tuple< _Elements > Class Template Reference	3330
5.714.1 Detailed Description	3332

5.715	<code>std::tuple< _T1 ></code> Class Template Reference	3332
5.715.1	Detailed Description	3332
5.716	<code>std::tuple< _T1, _T2 ></code> Class Template Reference	3333
5.716.1	Detailed Description	3333
5.717	<code>std::tuple_element< 0, tuple< _Head, _Tail...> ></code> Struct Template Reference	3334
5.717.1	Detailed Description	3334
5.718	<code>std::tuple_element< __i, tuple< _Head, _Tail...> ></code> Struct Template Reference	3334
5.718.1	Detailed Description	3334
5.719	<code>std::tuple_size< tuple< _Elements...> ></code> Struct Template Reference	3335
5.719.1	Detailed Description	3335
5.720	<code>std::type_index</code> Struct Reference	3335
5.720.1	Detailed Description	3336
5.721	<code>std::type_info</code> Class Reference	3336
5.721.1	Detailed Description	3337
5.721.2	Constructor & Destructor Documentation	3337
5.721.3	Member Function Documentation	3337
5.722	<code>std::unary_function< _Arg, _Result ></code> Struct Template Reference	3338
5.722.1	Detailed Description	3339
5.722.2	Member Typedef Documentation	3339
5.723	<code>std::unary_negate< _Predicate ></code> Class Template Reference	3339
5.723.1	Detailed Description	3340
5.723.2	Member Typedef Documentation	3341
5.724	<code>std::underflow_error</code> Class Reference	3342
5.724.1	Detailed Description	3342
5.724.2	Member Function Documentation	3342
5.725	<code>std::uniform_int_distribution< _IntType ></code> Class Template Reference	3343
5.725.1	Detailed Description	3344
5.725.2	Member Typedef Documentation	3344
5.725.3	Constructor & Destructor Documentation	3344
5.725.4	Member Function Documentation	3344

5.726	std::uniform_int_distribution< _IntType >::param_type Struct Reference	3346
5.726.1	Detailed Description	3346
5.727	std::uniform_real_distribution< _RealType > Class Template Reference	3347
5.727.1	Detailed Description	3347
5.727.2	Member Typedef Documentation	3348
5.727.3	Constructor & Destructor Documentation	3348
5.727.4	Member Function Documentation	3348
5.728	std::uniform_real_distribution< _RealType >::param_type Struct Reference	3350
5.728.1	Detailed Description	3350
5.729	std::unique_lock< _Mutex > Class Template Reference	3351
5.729.1	Detailed Description	3352
5.730	std::unique_ptr< _Tp, _Dp > Class Template Reference	3352
5.730.1	Detailed Description	3353
5.731	std::unique_ptr< _Tp[], _Dp > Class Template Reference	3353
5.731.1	Detailed Description	3354
5.732	std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference	3355
5.732.1	Detailed Description	3357
5.733	std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference	3357
5.733.1	Detailed Description	3360
5.734	std::unordered_multiset< _Value, _Hash, _Pred, _Alloc > Class Template Reference	3360
5.734.1	Detailed Description	3362
5.735	std::unordered_set< _Value, _Hash, _Pred, _Alloc > Class Template Reference	3363
5.735.1	Detailed Description	3365
5.736	std::uses_allocator< _Tp, _Alloc > Struct Template Reference	3366
5.736.1	Detailed Description	3366
5.737	std::valarray< _Tp > Class Template Reference	3367
5.737.1	Detailed Description	3370

5.737.2 Constructor & Destructor Documentation	3370
5.738std::vector< _Tp, _Alloc > Class Template Reference	3370
5.738.1 Detailed Description	3374
5.738.2 Constructor & Destructor Documentation	3374
5.738.3 Member Function Documentation	3378
5.739std::vector< bool, _Alloc > Class Template Reference	3395
5.739.1 Detailed Description	3398
5.740std::weak_ptr< _Tp > Class Template Reference	3399
5.740.1 Detailed Description	3400
5.741std::weibull_distribution< _RealType > Class Template Reference . .	3400
5.741.1 Detailed Description	3401
5.741.2 Member Typedef Documentation	3401
5.741.3 Member Function Documentation	3401
5.742std::weibull_distribution< _RealType >::param_type Struct Reference	3403
5.742.1 Detailed Description	3403
6 File Documentation	3404
6.1 algo.h File Reference	3404
6.1.1 Detailed Description	3418
6.2 algbase.h File Reference	3418
6.2.1 Detailed Description	3420
6.3 algorithm File Reference	3420
6.3.1 Detailed Description	3420
6.4 algorithm File Reference	3420
6.4.1 Detailed Description	3422
6.5 algorithm File Reference	3422
6.5.1 Detailed Description	3422
6.6 algorithmfwd.h File Reference	3422
6.6.1 Detailed Description	3429
6.7 algorithmfwd.h File Reference	3429
6.7.1 Detailed Description	3441

6.8	allocator.h File Reference	3442
6.8.1	Detailed Description	3442
6.9	array File Reference	3443
6.9.1	Detailed Description	3443
6.10	array_allocator.h File Reference	3444
6.10.1	Detailed Description	3444
6.11	assoc_container.hpp File Reference	3444
6.11.1	Detailed Description	3446
6.12	atomic File Reference	3446
6.12.1	Detailed Description	3452
6.13	atomic_0.h File Reference	3452
6.13.1	Detailed Description	3453
6.14	atomic_2.h File Reference	3453
6.14.1	Detailed Description	3454
6.15	atomic_base.h File Reference	3454
6.15.1	Detailed Description	3456
6.16	atomic_word.h File Reference	3456
6.16.1	Detailed Description	3456
6.17	atomicity.h File Reference	3456
6.17.1	Detailed Description	3457
6.18	auto_ptr.h File Reference	3457
6.18.1	Detailed Description	3458
6.19	backward_warning.h File Reference	3458
6.19.1	Detailed Description	3458
6.20	balanced_quicksort.h File Reference	3458
6.20.1	Detailed Description	3459
6.21	base.h File Reference	3459
6.21.1	Detailed Description	3461
6.22	base.h File Reference	3461
6.22.1	Detailed Description	3461
6.23	basic_file.h File Reference	3461

6.23.1 Detailed Description	3461
6.24 basic_ios.h File Reference	3461
6.24.1 Detailed Description	3462
6.25 basic_ios.tcc File Reference	3462
6.25.1 Detailed Description	3462
6.26 basic_iterator.h File Reference	3462
6.26.1 Detailed Description	3463
6.27 basic_string.h File Reference	3463
6.27.1 Detailed Description	3467
6.28 basic_string.tcc File Reference	3467
6.28.1 Detailed Description	3467
6.29 basic_types.hpp File Reference	3468
6.29.1 Detailed Description	3468
6.30 binders.h File Reference	3468
6.30.1 Detailed Description	3469
6.31 bitmap_allocator.h File Reference	3469
6.31.1 Detailed Description	3470
6.31.2 Define Documentation	3470
6.32 bitset File Reference	3471
6.32.1 Detailed Description	3472
6.33 bitset File Reference	3472
6.33.1 Detailed Description	3473
6.34 bitset File Reference	3473
6.34.1 Detailed Description	3474
6.35 boost_concept_check.h File Reference	3474
6.35.1 Detailed Description	3474
6.36 c++0x_warning.h File Reference	3475
6.36.1 Detailed Description	3475
6.37 c++allocator.h File Reference	3475
6.37.1 Detailed Description	3475
6.38 c++config.h File Reference	3475

6.38.1 Detailed Description	3480
6.39 c++io.h File Reference	3480
6.39.1 Detailed Description	3480
6.40 c++locale.h File Reference	3481
6.40.1 Detailed Description	3481
6.41 c++locale_internal.h File Reference	3481
6.41.1 Detailed Description	3481
6.42 cassert File Reference	3481
6.42.1 Detailed Description	3481
6.43 cast.h File Reference	3482
6.43.1 Detailed Description	3482
6.44 ccomplex File Reference	3483
6.44.1 Detailed Description	3483
6.45 ccomplex File Reference	3483
6.45.1 Detailed Description	3483
6.46 cctype File Reference	3483
6.46.1 Detailed Description	3483
6.47 cctype File Reference	3483
6.47.1 Detailed Description	3483
6.48 cerrno File Reference	3484
6.48.1 Detailed Description	3484
6.49 cenv File Reference	3484
6.49.1 Detailed Description	3484
6.50 cenv File Reference	3484
6.50.1 Detailed Description	3484
6.51 cfloat File Reference	3484
6.51.1 Detailed Description	3484
6.52 cfloat File Reference	3485
6.52.1 Detailed Description	3485
6.53 char_traits.h File Reference	3485
6.53.1 Detailed Description	3486

6.54	checkers.h File Reference	3486
6.54.1	Detailed Description	3486
6.55	chrono File Reference	3486
6.55.1	Detailed Description	3489
6.56	cinttypes File Reference	3489
6.56.1	Detailed Description	3489
6.57	cinttypes File Reference	3490
6.57.1	Detailed Description	3490
6.58	ciso646 File Reference	3490
6.58.1	Detailed Description	3490
6.59	climits File Reference	3490
6.59.1	Detailed Description	3490
6.60	climits File Reference	3490
6.60.1	Detailed Description	3491
6.61	locale File Reference	3491
6.61.1	Detailed Description	3491
6.62	cmath File Reference	3491
6.62.1	Detailed Description	3494
6.63	cmath File Reference	3494
6.63.1	Detailed Description	3497
6.64	codecvt.h File Reference	3497
6.64.1	Detailed Description	3498
6.65	codecvt_specializations.h File Reference	3498
6.65.1	Detailed Description	3499
6.66	compatibility.h File Reference	3499
6.66.1	Detailed Description	3499
6.67	compatibility.h File Reference	3499
6.67.1	Detailed Description	3500
6.68	compiletime_settings.h File Reference	3500
6.68.1	Detailed Description	3500
6.68.2	Define Documentation	3500

6.69	complex File Reference	3502
6.69.1	Detailed Description	3507
6.70	complex File Reference	3507
6.70.1	Detailed Description	3508
6.71	complex.h File Reference	3509
6.71.1	Detailed Description	3509
6.72	concept_check.h File Reference	3509
6.72.1	Detailed Description	3509
6.73	concurrency.h File Reference	3509
6.73.1	Detailed Description	3510
6.74	cond_dealtor.hpp File Reference	3510
6.74.1	Detailed Description	3510
6.75	condition_variable File Reference	3510
6.75.1	Detailed Description	3511
6.76	constructors_destructor_fn_imps.hpp File Reference	3511
6.76.1	Detailed Description	3512
6.77	container_base_dispatch.hpp File Reference	3512
6.77.1	Detailed Description	3512
6.78	cpp_type_traits.h File Reference	3512
6.78.1	Detailed Description	3512
6.79	cpu_defines.h File Reference	3512
6.79.1	Detailed Description	3512
6.80	csetjmp File Reference	3513
6.80.1	Detailed Description	3513
6.81	csignal File Reference	3513
6.81.1	Detailed Description	3513
6.82	cstdarg File Reference	3513
6.82.1	Detailed Description	3514
6.83	cstdarg File Reference	3514
6.83.1	Detailed Description	3514
6.84	cstdbool File Reference	3514

6.84.1 Detailed Description	3514
6.85 cstdbool File Reference	3514
6.85.1 Detailed Description	3514
6.86 cstddef File Reference	3515
6.86.1 Detailed Description	3515
6.87 cstdint File Reference	3515
6.87.1 Detailed Description	3515
6.88 cstdint File Reference	3515
6.88.1 Detailed Description	3515
6.89 cstdio File Reference	3515
6.89.1 Detailed Description	3516
6.90 cstdio File Reference	3516
6.90.1 Detailed Description	3516
6.91 cstdlib File Reference	3516
6.91.1 Detailed Description	3516
6.92 cstdlib File Reference	3517
6.92.1 Detailed Description	3517
6.93 cstring File Reference	3517
6.93.1 Detailed Description	3517
6.94 ctgmath File Reference	3517
6.94.1 Detailed Description	3517
6.95 ctgmath File Reference	3518
6.95.1 Detailed Description	3518
6.96 ctime File Reference	3518
6.96.1 Detailed Description	3518
6.97 ctime File Reference	3518
6.97.1 Detailed Description	3518
6.98 ctype_base.h File Reference	3518
6.98.1 Detailed Description	3519
6.99 ctype_inline.h File Reference	3519
6.99.1 Detailed Description	3519

6.100ctype_noninline.h File Reference	3519
6.100.1 Detailed Description	3519
6.101cwchar File Reference	3519
6.101.1 Detailed Description	3520
6.102cwchar File Reference	3520
6.102.1 Detailed Description	3520
6.103cwctype File Reference	3520
6.103.1 Detailed Description	3520
6.104cwctype File Reference	3521
6.104.1 Detailed Description	3521
6.105cxxabi.h File Reference	3521
6.105.1 Detailed Description	3522
6.106cxxabi_forced.h File Reference	3523
6.106.1 Detailed Description	3523
6.107cxxabi_tweaks.h File Reference	3523
6.107.1 Detailed Description	3523
6.108debug.h File Reference	3524
6.108.1 Detailed Description	3524
6.109debug_allocator.h File Reference	3524
6.109.1 Detailed Description	3525
6.110debug_map_base.hpp File Reference	3525
6.110.1 Detailed Description	3525
6.111decimal File Reference	3525
6.111.1 Detailed Description	3537
6.112deque File Reference	3537
6.112.1 Detailed Description	3537
6.113deque File Reference	3537
6.113.1 Detailed Description	3538
6.114deque File Reference	3538
6.114.1 Detailed Description	3539
6.115deque.tcc File Reference	3539

6.115.1 Detailed Description	3540
6.116enc_filebuf.h File Reference	3540
6.116.1 Detailed Description	3540
6.117equally_split.h File Reference	3541
6.117.1 Detailed Description	3541
6.118error_constants.h File Reference	3541
6.118.1 Detailed Description	3542
6.119exception File Reference	3542
6.119.1 Detailed Description	3543
6.120exception.hpp File Reference	3543
6.120.1 Detailed Description	3543
6.121exception_defines.h File Reference	3544
6.121.1 Detailed Description	3544
6.122exception_ptr.h File Reference	3544
6.122.1 Detailed Description	3545
6.123extc++.h File Reference	3545
6.123.1 Detailed Description	3545
6.124extptr_allocator.h File Reference	3545
6.124.1 Detailed Description	3545
6.125features.h File Reference	3546
6.125.1 Detailed Description	3546
6.125.2 Define Documentation	3546
6.126fenv.h File Reference	3549
6.126.1 Detailed Description	3549
6.127find.h File Reference	3549
6.127.1 Detailed Description	3549
6.128find_selectors.h File Reference	3550
6.128.1 Detailed Description	3550
6.129for_each.h File Reference	3550
6.129.1 Detailed Description	3551
6.130for_each_selectors.h File Reference	3551

6.130.1 Detailed Description	3553
6.131formatter.h File Reference	3553
6.131.1 Detailed Description	3554
6.132forward_list File Reference	3554
6.132.1 Detailed Description	3554
6.133forward_list File Reference	3554
6.133.1 Detailed Description	3555
6.134forward_list File Reference	3555
6.134.1 Detailed Description	3556
6.135forward_list.h File Reference	3556
6.135.1 Detailed Description	3558
6.136forward_list.tcc File Reference	3558
6.136.1 Detailed Description	3558
6.137fstream File Reference	3558
6.137.1 Detailed Description	3559
6.138fstream.tcc File Reference	3559
6.138.1 Detailed Description	3559
6.139functexcept.h File Reference	3559
6.139.1 Detailed Description	3560
6.140functional File Reference	3560
6.140.1 Detailed Description	3565
6.141functional File Reference	3565
6.141.1 Detailed Description	3567
6.142functional_hash.h File Reference	3567
6.142.1 Detailed Description	3567
6.143functions.h File Reference	3567
6.143.1 Detailed Description	3570
6.144future File Reference	3570
6.144.1 Detailed Description	3572
6.145gslice.h File Reference	3572
6.145.1 Detailed Description	3573

6.146	gslice_array.h File Reference	3573
6.146.1	Detailed Description	3573
6.147	hash_bytes.h File Reference	3573
6.147.1	Detailed Description	3574
6.148	hash_fun.h File Reference	3574
6.148.1	Detailed Description	3574
6.149	hash_map File Reference	3574
6.149.1	Detailed Description	3575
6.150	hash_policy.hpp File Reference	3575
6.150.1	Detailed Description	3576
6.151	hash_set File Reference	3576
6.151.1	Detailed Description	3577
6.152	hashtable.h File Reference	3577
6.152.1	Detailed Description	3578
6.153	hashtable.h File Reference	3578
6.153.1	Detailed Description	3578
6.154	hashtable_policy.h File Reference	3578
6.154.1	Detailed Description	3579
6.155	indirect_array.h File Reference	3579
6.155.1	Detailed Description	3580
6.156	initializer_list File Reference	3580
6.156.1	Detailed Description	3580
6.157	iomanip File Reference	3580
6.157.1	Detailed Description	3582
6.158	ios File Reference	3582
6.158.1	Detailed Description	3582
6.159	ios_base.h File Reference	3582
6.159.1	Detailed Description	3584
6.160	iosfwd File Reference	3585
6.160.1	Detailed Description	3585
6.161	iostream File Reference	3586

6.161.1 Detailed Description	3586
6.162istream File Reference	3586
6.162.1 Detailed Description	3588
6.163istream.tcc File Reference	3588
6.163.1 Detailed Description	3588
6.164iterator File Reference	3588
6.164.1 Detailed Description	3588
6.165iterator File Reference	3589
6.165.1 Detailed Description	3589
6.166iterator.h File Reference	3589
6.166.1 Detailed Description	3590
6.167iterator_tracker.h File Reference	3590
6.167.1 Detailed Description	3591
6.168limits File Reference	3592
6.168.1 Detailed Description	3594
6.169list File Reference	3594
6.169.1 Detailed Description	3594
6.170list File Reference	3594
6.170.1 Detailed Description	3595
6.171list File Reference	3595
6.171.1 Detailed Description	3596
6.172list.tcc File Reference	3596
6.172.1 Detailed Description	3596
6.173list_partition.h File Reference	3596
6.173.1 Detailed Description	3597
6.174list_update_policy.hpp File Reference	3597
6.174.1 Detailed Description	3597
6.175locale File Reference	3598
6.175.1 Detailed Description	3598
6.176locale_classes.h File Reference	3598
6.176.1 Detailed Description	3599

6.177locale_classes.tcc File Reference	3599
6.177.1 Detailed Description	3599
6.178locale_facets.h File Reference	3599
6.178.1 Detailed Description	3602
6.179locale_facets.tcc File Reference	3602
6.179.1 Detailed Description	3602
6.180locale_facets_nonio.h File Reference	3602
6.180.1 Detailed Description	3604
6.181locale_facets_nonio.tcc File Reference	3604
6.181.1 Detailed Description	3604
6.182localefwd.h File Reference	3604
6.182.1 Detailed Description	3605
6.183losertree.h File Reference	3605
6.183.1 Detailed Description	3607
6.184macros.h File Reference	3607
6.184.1 Detailed Description	3608
6.184.2 Define Documentation	3608
6.185malloc_allocator.h File Reference	3611
6.185.1 Detailed Description	3611
6.186map File Reference	3612
6.186.1 Detailed Description	3612
6.187map File Reference	3612
6.187.1 Detailed Description	3612
6.188map File Reference	3612
6.188.1 Detailed Description	3612
6.189map.h File Reference	3612
6.189.1 Detailed Description	3613
6.190map.h File Reference	3613
6.190.1 Detailed Description	3614
6.191mask_array.h File Reference	3614
6.191.1 Detailed Description	3615

6.192memory File Reference	3615
6.192.1 Detailed Description	3615
6.193memory File Reference	3615
6.193.1 Detailed Description	3616
6.194merge.h File Reference	3616
6.194.1 Detailed Description	3617
6.195messages_members.h File Reference	3617
6.195.1 Detailed Description	3617
6.196move.h File Reference	3617
6.196.1 Detailed Description	3618
6.197mt_allocator.h File Reference	3618
6.197.1 Detailed Description	3619
6.198multimap.h File Reference	3619
6.198.1 Detailed Description	3620
6.199multimap.h File Reference	3621
6.199.1 Detailed Description	3622
6.200multiseq_selection.h File Reference	3622
6.200.1 Detailed Description	3623
6.201multiset.h File Reference	3623
6.201.1 Detailed Description	3624
6.202multiset.h File Reference	3624
6.202.1 Detailed Description	3625
6.203multiway_merge.h File Reference	3625
6.203.1 Detailed Description	3630
6.203.2 Define Documentation	3630
6.204multiway_mergesort.h File Reference	3630
6.204.1 Detailed Description	3631
6.205mutex File Reference	3632
6.205.1 Detailed Description	3633
6.206nested_exception.h File Reference	3633
6.206.1 Detailed Description	3634

6.207new File Reference	3634
6.207.1 Detailed Description	3635
6.207.2 Function Documentation	3635
6.208new_allocator.h File Reference	3639
6.208.1 Detailed Description	3640
6.209numeric File Reference	3640
6.209.1 Detailed Description	3640
6.210numeric File Reference	3640
6.210.1 Detailed Description	3641
6.211numeric File Reference	3641
6.211.1 Detailed Description	3644
6.212numeric_traits.h File Reference	3644
6.212.1 Detailed Description	3645
6.213numeric_fwd.h File Reference	3645
6.213.1 Detailed Description	3647
6.214omp_loop.h File Reference	3647
6.214.1 Detailed Description	3648
6.215omp_loop_static.h File Reference	3648
6.215.1 Detailed Description	3648
6.216os_defines.h File Reference	3649
6.216.1 Detailed Description	3649
6.217ostream File Reference	3649
6.217.1 Detailed Description	3650
6.218ostream.tcc File Reference	3651
6.218.1 Detailed Description	3651
6.219ostream_insert.h File Reference	3651
6.219.1 Detailed Description	3652
6.220par_loop.h File Reference	3652
6.220.1 Detailed Description	3652
6.221parallel.h File Reference	3652
6.221.1 Detailed Description	3652

6.222	partial_sum.h File Reference	3653
6.222.1	Detailed Description	3653
6.223	partition.h File Reference	3653
6.223.1	Detailed Description	3654
6.223.2	Define Documentation	3654
6.224	pod_char_traits.h File Reference	3654
6.224.1	Detailed Description	3655
6.225	pointer.h File Reference	3655
6.225.1	Detailed Description	3658
6.226	pool_allocator.h File Reference	3658
6.226.1	Detailed Description	3659
6.227	postypes.h File Reference	3659
6.227.1	Detailed Description	3659
6.228	priority_queue.hpp File Reference	3660
6.228.1	Detailed Description	3660
6.229	priority_queue_base_dispatch.hpp File Reference	3660
6.229.1	Detailed Description	3660
6.230	profiler.h File Reference	3660
6.230.1	Detailed Description	3663
6.231	profiler_algos.h File Reference	3663
6.231.1	Detailed Description	3664
6.232	profiler_container_size.h File Reference	3664
6.232.1	Detailed Description	3665
6.233	profiler_hash_func.h File Reference	3665
6.233.1	Detailed Description	3665
6.234	profiler_hashtable_size.h File Reference	3666
6.234.1	Detailed Description	3666
6.235	profiler_list_to_slist.h File Reference	3666
6.235.1	Detailed Description	3667
6.236	profiler_list_to_vector.h File Reference	3667
6.236.1	Detailed Description	3668

6.237	profiler_map_to_unordered_map.h File Reference	3668
6.237.1	Detailed Description	3669
6.238	profiler_node.h File Reference	3669
6.238.1	Detailed Description	3670
6.239	profiler_state.h File Reference	3670
6.239.1	Detailed Description	3670
6.240	profiler_trace.h File Reference	3671
6.240.1	Detailed Description	3673
6.241	profiler_vector_size.h File Reference	3673
6.241.1	Detailed Description	3674
6.242	profiler_vector_to_list.h File Reference	3674
6.242.1	Detailed Description	3675
6.243	queue File Reference	3675
6.243.1	Detailed Description	3675
6.244	queue.h File Reference	3675
6.244.1	Detailed Description	3675
6.244.2	Define Documentation	3676
6.245	quicksort.h File Reference	3676
6.245.1	Detailed Description	3676
6.246	random File Reference	3677
6.246.1	Detailed Description	3677
6.247	random.h File Reference	3677
6.247.1	Detailed Description	3684
6.248	random.tcc File Reference	3684
6.248.1	Detailed Description	3689
6.249	random_number.h File Reference	3689
6.249.1	Detailed Description	3689
6.250	random_shuffle.h File Reference	3690
6.250.1	Detailed Description	3691
6.251	range_access.h File Reference	3691
6.251.1	Detailed Description	3691

6.252ratio File Reference	3692
6.252.1 Detailed Description	3692
6.253rb_tree File Reference	3692
6.253.1 Detailed Description	3693
6.254rc_string_base.h File Reference	3693
6.254.1 Detailed Description	3693
6.255regex File Reference	3693
6.255.1 Detailed Description	3693
6.256regex.h File Reference	3693
6.256.1 Detailed Description	3699
6.257regex_compiler.h File Reference	3700
6.257.1 Detailed Description	3700
6.258regex_constants.h File Reference	3700
6.258.1 Detailed Description	3701
6.259regex_cursor.h File Reference	3701
6.259.1 Detailed Description	3702
6.260regex_error.h File Reference	3702
6.260.1 Detailed Description	3703
6.261regex_grep_matcher.h File Reference	3703
6.261.1 Detailed Description	3703
6.262regex_grep_matcher.tcc File Reference	3704
6.262.1 Detailed Description	3704
6.263regex_nfa.h File Reference	3704
6.263.1 Detailed Description	3705
6.264regex_nfa.tcc File Reference	3705
6.264.1 Detailed Description	3705
6.265rope File Reference	3705
6.265.1 Detailed Description	3709
6.266ropeimpl.h File Reference	3709
6.266.1 Detailed Description	3710
6.267safe_base.h File Reference	3710

6.267.1 Detailed Description	3711
6.268safe_iterator.h File Reference	3711
6.268.1 Detailed Description	3712
6.269safe_iterator.tcc File Reference	3712
6.269.1 Detailed Description	3713
6.270safe_sequence.h File Reference	3713
6.270.1 Detailed Description	3713
6.271safe_sequence.tcc File Reference	3713
6.271.1 Detailed Description	3713
6.272search.h File Reference	3713
6.272.1 Detailed Description	3714
6.273set File Reference	3714
6.273.1 Detailed Description	3714
6.274set File Reference	3714
6.274.1 Detailed Description	3714
6.275set File Reference	3714
6.275.1 Detailed Description	3714
6.276set.h File Reference	3715
6.276.1 Detailed Description	3715
6.277set.h File Reference	3716
6.277.1 Detailed Description	3716
6.278set_operations.h File Reference	3717
6.278.1 Detailed Description	3717
6.279settings.h File Reference	3718
6.279.1 Detailed Description	3718
6.279.2 parallelization_decision	3718
6.279.3 Define Documentation	3719
6.280shared_ptr.h File Reference	3719
6.280.1 Detailed Description	3721
6.281shared_ptr_base.h File Reference	3721
6.281.1 Detailed Description	3723

6.282slice_array.h File Reference	3723
6.282.1 Detailed Description	3723
6.283slist File Reference	3724
6.283.1 Detailed Description	3725
6.284sort.h File Reference	3725
6.284.1 Detailed Description	3726
6.285sso_string_base.h File Reference	3726
6.285.1 Detailed Description	3726
6.286sstream File Reference	3726
6.286.1 Detailed Description	3727
6.287sstream.tcc File Reference	3727
6.287.1 Detailed Description	3727
6.288stack File Reference	3727
6.288.1 Detailed Description	3727
6.289standard_policies.hpp File Reference	3728
6.289.1 Detailed Description	3728
6.290stdc++.h File Reference	3728
6.290.1 Detailed Description	3728
6.291stdexcept File Reference	3728
6.291.1 Detailed Description	3729
6.292stdio_filebuf.h File Reference	3729
6.292.1 Detailed Description	3729
6.293stdio_sync_filebuf.h File Reference	3729
6.293.1 Detailed Description	3730
6.294stdtr1c++.h File Reference	3730
6.294.1 Detailed Description	3730
6.295stl_algo.h File Reference	3730
6.295.1 Detailed Description	3743
6.296stl_algo.h File Reference	3743
6.296.1 Detailed Description	3746
6.297stl_bvector.h File Reference	3746

6.297.1 Detailed Description	3747
6.298stl_construct.h File Reference	3747
6.298.1 Detailed Description	3748
6.299stl_deque.h File Reference	3748
6.299.1 Detailed Description	3751
6.299.2 Define Documentation	3751
6.300stl_function.h File Reference	3752
6.300.1 Detailed Description	3754
6.301stl_heap.h File Reference	3754
6.301.1 Detailed Description	3756
6.302stl_iterator.h File Reference	3756
6.302.1 Detailed Description	3760
6.303stl_iterator_base_funcs.h File Reference	3761
6.303.1 Detailed Description	3762
6.304stl_iterator_base_types.h File Reference	3762
6.304.1 Detailed Description	3763
6.305stl_list.h File Reference	3763
6.305.1 Detailed Description	3764
6.306stl_map.h File Reference	3765
6.306.1 Detailed Description	3765
6.307stl_multimap.h File Reference	3766
6.307.1 Detailed Description	3766
6.308stl_multiset.h File Reference	3767
6.308.1 Detailed Description	3767
6.309stl_numeric.h File Reference	3768
6.309.1 Detailed Description	3768
6.310stl_pair.h File Reference	3769
6.310.1 Detailed Description	3770
6.311stl_queue.h File Reference	3770
6.311.1 Detailed Description	3771
6.312stl_raw_storage_iter.h File Reference	3771

6.312.1 Detailed Description	3771
6.313stl_relops.h File Reference	3771
6.313.1 Detailed Description	3772
6.314stl_set.h File Reference	3772
6.314.1 Detailed Description	3773
6.315stl_stack.h File Reference	3773
6.315.1 Detailed Description	3774
6.316stl_tempbuf.h File Reference	3774
6.316.1 Detailed Description	3774
6.317stl_tree.h File Reference	3775
6.317.1 Detailed Description	3776
6.318stl_uninitialized.h File Reference	3776
6.318.1 Detailed Description	3778
6.319stl_vector.h File Reference	3778
6.319.1 Detailed Description	3779
6.320stream_iterator.h File Reference	3780
6.320.1 Detailed Description	3780
6.321streambuf File Reference	3780
6.321.1 Detailed Description	3781
6.322streambuf.tcc File Reference	3781
6.322.1 Detailed Description	3781
6.323streambuf_iterator.h File Reference	3782
6.323.1 Detailed Description	3783
6.324string File Reference	3783
6.324.1 Detailed Description	3783
6.325string File Reference	3783
6.325.1 Detailed Description	3785
6.326string_conversions.h File Reference	3786
6.326.1 Detailed Description	3786
6.327stringfwd.h File Reference	3786
6.327.1 Detailed Description	3786

6.328	stringstream File Reference	3787
6.328.1	Detailed Description	3787
6.329	system_error File Reference	3787
6.329.1	Detailed Description	3788
6.330	tag_and_trait.hpp File Reference	3788
6.330.1	Detailed Description	3790
6.331	tags.h File Reference	3790
6.331.1	Detailed Description	3792
6.332	tgmath.h File Reference	3792
6.332.1	Detailed Description	3792
6.333	thread File Reference	3792
6.333.1	Detailed Description	3793
6.334	throw_allocator.h File Reference	3793
6.334.1	Detailed Description	3796
6.335	time_members.h File Reference	3796
6.335.1	Detailed Description	3796
6.336	tree_policy.hpp File Reference	3796
6.336.1	Detailed Description	3797
6.337	tree_trace_base.hpp File Reference	3797
6.337.1	Detailed Description	3797
6.338	trie_policy.hpp File Reference	3797
6.338.1	Detailed Description	3797
6.339	tuple File Reference	3798
6.339.1	Detailed Description	3800
6.340	type_traits File Reference	3800
6.340.1	Detailed Description	3806
6.341	type_traits.h File Reference	3806
6.341.1	Detailed Description	3806
6.342	type_utils.hpp File Reference	3806
6.342.1	Detailed Description	3807
6.343	typeid File Reference	3807

6.343.1 Detailed Description	3807
6.344typeinfo File Reference	3808
6.344.1 Detailed Description	3808
6.345typelist.h File Reference	3808
6.345.1 Detailed Description	3809
6.346types.h File Reference	3810
6.346.1 Detailed Description	3810
6.347types_traits.hpp File Reference	3811
6.347.1 Detailed Description	3811
6.348unique_copy.h File Reference	3811
6.348.1 Detailed Description	3811
6.349unique_ptr.h File Reference	3811
6.349.1 Detailed Description	3813
6.350unordered_map File Reference	3813
6.350.1 Detailed Description	3813
6.351unordered_map File Reference	3813
6.351.1 Detailed Description	3814
6.352unordered_map File Reference	3814
6.352.1 Detailed Description	3816
6.353unordered_map.h File Reference	3816
6.353.1 Detailed Description	3817
6.354unordered_set File Reference	3818
6.354.1 Detailed Description	3818
6.355unordered_set File Reference	3818
6.355.1 Detailed Description	3819
6.356unordered_set File Reference	3819
6.356.1 Detailed Description	3820
6.357unordered_set.h File Reference	3820
6.357.1 Detailed Description	3821
6.358utility File Reference	3822
6.358.1 Detailed Description	3822

6.359valarray File Reference	3822
6.359.1 Detailed Description	3828
6.360valarray_after.h File Reference	3828
6.360.1 Detailed Description	3842
6.361valarray_array.h File Reference	3842
6.361.1 Detailed Description	3853
6.362valarray_array.tcc File Reference	3853
6.362.1 Detailed Description	3854
6.363valarray_before.h File Reference	3854
6.363.1 Detailed Description	3854
6.364vector File Reference	3854
6.364.1 Detailed Description	3854
6.365vector File Reference	3855
6.365.1 Detailed Description	3855
6.366vector File Reference	3856
6.366.1 Detailed Description	3857
6.367vector.tcc File Reference	3857
6.367.1 Detailed Description	3857
6.368vstring.h File Reference	3857
6.368.1 Detailed Description	3862
6.369vstring.tcc File Reference	3862
6.369.1 Detailed Description	3863
6.370vstring_fwd.h File Reference	3863
6.370.1 Detailed Description	3864
6.371vstring_util.h File Reference	3864
6.371.1 Detailed Description	3864
6.372workstealing.h File Reference	3864
6.372.1 Detailed Description	3865

1 Todo List

Member `__glibcxx_check_insert_range(_Position, _First, _Last)` We would like to be able to check for noninterference of `_Position` and the range `[_First, _Last)`, but that can't (in general) be done.

Member `__glibcxx_check_insert_range_after(_Position, _First, _Last)` We would like to be able to check for noninterference of `_Position` and the range `[_First, _Last)`, but that can't (in general) be done.

Member `__gnu_cxx::distance(_InputIterator __first, _InputIterator __last, _Distance &__n)`
Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Class `__gnu_cxx::hash_map<_Key, _Tp, _HashFn, _EqualKey, _Alloc>`
Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Class `__gnu_cxx::hash_multimap<_Key, _Tp, _HashFn, _EqualKey, _Alloc>`
Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Class `__gnu_cxx::hash_multiset<_Value, _HashFcn, _EqualKey, _Alloc>`
Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Class `__gnu_cxx::hash_set<_Value, _HashFcn, _EqualKey, _Alloc>` Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Member `__gnu_cxx::iota(_ForwardIter __first, _ForwardIter __last, _Tp __value)`
Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Member `__gnu_cxx::is_heap(_RandomAccessIterator __first, _RandomAccessIterator __last)`
Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Member [`__gnu_cxx::is_heap`](#)(`_RandomAccessIterator __first`, `_RandomAccessIterator __last`, `_StrictWeakOrdering __comp`)

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_10.html

Member [`__gnu_cxx::is_sorted`](#)(`_ForwardIterator __first`, `_ForwardIterator __last`)

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_10.html

Member [`__gnu_cxx::is_sorted`](#)(`_ForwardIterator __first`, `_ForwardIterator __last`, `_StrictWeakOrdering __comp`)

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_10.html

Member [`__gnu_cxx::power`](#)(`_Tp __x`, `_Integer __n`, `_MonoidOperation __monoid_op`)

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_10.html

Member [`__gnu_cxx::power`](#)(`_Tp __x`, `_Integer __n`) Needs documentation! See

http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_10.html

Member [`__gnu_cxx::random_sample`](#)(`_InputIterator __first`, `_InputIterator __last`, `_RandomAccessIterator __out`)

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_10.html

Member [`__gnu_cxx::random_sample`](#)(`_InputIterator __first`, `_InputIterator __last`, `_RandomAccessIterator __out`)

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_10.html

Member [`__gnu_cxx::random_sample_n`](#)(`_ForwardIterator __first`, `_ForwardIterator __last`, `_OutputIterator __out`)

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_10.html

Member [`__gnu_cxx::random_sample_n`](#)(`_ForwardIterator __first`, `_ForwardIterator __last`, `_OutputIterator __out`)

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_10.html

Class `__gnu_cxx::rb_tree<_Key, _Value, _KeyOfValue, _Compare, _Alloc>`

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Class `__gnu_cxx::rope<_CharT, _Alloc>` Needs documentation! See

http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Class `__gnu_cxx::slist<_Tp, _Alloc>` Needs documentation! See

http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Member `__gnu_debug::_Safe_iterator::operator->() const` Make this correct

w.r.t. iterators that return proxies

Use `addressof()` instead of `&` operator

Class `std::basic_string<_CharT, _Traits, _Alloc>` Needs documentation! See

http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Member `std::match_results::format(_Out_iter __out, const string_type &__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)`

Implement this function.

Member `std::match_results::format(const string_type &__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)`

Implement this function.

Member `std::operator==(const match_results<_Bi_iter, _Allocator> &__m1, const match_results<_Bi_iter, _Allocator> &__m2)`

Implement this function.

Member `std::regex_iterator::operator!=(const regex_iterator &__rhs)`

Implement this function.

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Member `std::regex_iterator::operator*()` Implement this function.

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Member **`std::regex_iterator::operator++()`** Implement this function.

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentati
style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentati
style.html)

Member **`std::regex_iterator::operator++(int)`** Implement this function.

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentati
style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentati
style.html)

Member **`std::regex_iterator::operator->()`** Implement this function.

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentati
style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentati
style.html)

Member **`std::regex_iterator::operator=(const regex_iterator &__rhs)`**

Implement this function.

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentati
style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentati
style.html)

Member **`std::regex_iterator::operator==(const regex_iterator &__rhs)`**

Implement this function.

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentati
style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentati
style.html)

Member **`std::regex_iterator::regex_iterator()`** Implement this function.

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentati
style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentati
style.html)

Member **`std::regex_iterator::regex_iterator(_Bi_iter __a, _Bi_iter __b, const regex_type &__re, regex_constants::`**

Implement this function.

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentati
style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentati
style.html)

Member **`std::regex_iterator::regex_iterator(const regex_iterator &__rhs)`**

Implement this function.

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentati
style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentati
style.html)

Member **std::regex_match**(_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, const basic_string< _Ch_type, _String_allocator > &__s, const basic_regex_constants< _Ch_type, _Rx_traits > &__c)
Implement this function.

Member **std::regex_replace**(_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, const basic_string< _Ch_type, _String_allocator > &__s, const basic_regex_constants< _Ch_type, _Rx_traits > &__c)
Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentati.html>
style.html
Implement this function.
Implement this function.

Member **std::regex_replace**(const basic_string< _Ch_type > &__s, const basic_regex< _Ch_type, _Rx_traits > &__re, const basic_string< _Ch_type, _String_allocator > &__s, const basic_regex_constants< _Ch_type, _Rx_traits > &__c)
Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentati.html>
style.html

Member **std::regex_search**(_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, match_results< _Bi_iter, _Allocator > &__m, const basic_string< _Ch_type, _String_allocator > &__s, const basic_regex_constants< _Ch_type, _Rx_traits > &__c)
Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentati.html>
style.html

Member **std::regex_search**(const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits > &__re, match_results< const _Ch_type *, _Allocator > &__m, const basic_string< _Ch_type, _String_allocator > &__s, const basic_regex_constants< _Ch_type, _Rx_traits > &__c)
Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentati.html>
style.html

Member **std::regex_search**(const basic_string< _Ch_type, _Ch_traits, _String_allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__re, match_results< const _Ch_type *, _Allocator > &__m, const basic_string< _Ch_type, _String_allocator > &__s, const basic_regex_constants< _Ch_type, _Rx_traits > &__c)
Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentati.html>
style.html

Member **std::regex_search**(const _Ch_type *__s, match_results< const _Ch_type *, _Allocator > &__m, const basic_string< _Ch_type, _String_allocator > &__s, const basic_regex_constants< _Ch_type, _Rx_traits > &__c)
Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentati.html>
style.html

Member **std::regex_search**(_Bi_iter __first, _Bi_iter __last, match_results< _Bi_iter, _Allocator > &__m, const basic_string< _Ch_type, _String_allocator > &__s, const basic_regex_constants< _Ch_type, _Rx_traits > &__c)
Implement this function.

Member **std::regex_token_iterator::operator!=(const regex_token_iterator &__rhs)**
Implement this function.

Member **std::regex_token_iterator::operator*()** Implement this function.

Member `std::regex_token_iterator::operator++()` Implement this function.

Member `std::regex_token_iterator::operator++(int)` Implement this function.

Member `std::regex_token_iterator::operator->()` Implement this function.

Member `std::regex_token_iterator::operator=(const regex_token_iterator &__rhs)`
Implement this function.

Member `std::regex_token_iterator::operator==(const regex_token_iterator &__rhs)`
Implement this function.

Member `std::regex_token_iterator::regex_token_iterator(_Bi_iter __a, _Bi_iter __b, const regex_type &__re, int __sub)`
Implement this function.

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentati.html>
[style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentati.html)

Member `std::regex_token_iterator::regex_token_iterator(_Bi_iter __a, _Bi_iter __b, const regex_type &__re, const regex_token_iterator &__rhs)`
Implement this function.

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentati.html>
[style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentati.html)

Member `std::regex_token_iterator::regex_token_iterator(const regex_token_iterator &__rhs)`
Implement this function.

Member `std::regex_token_iterator::regex_token_iterator(_Bi_iter __a, _Bi_iter __b, const regex_type &__re, const regex_token_iterator &__rhs)`
Implement this function.

Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentati.html>
[style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentati.html)

Member `std::regex_token_iterator::regex_token_iterator()` Implement this function.

Member `std::regex_traits::lookup_classname(_Fwd_iter __first, _Fwd_iter __last, bool __icase=false) const`
Implement this function.

Member **`std::regex_traits::lookup_collatename(_Fwd_iter __first, _Fwd_iter __last) const`**
Implement this function.

Member **`std::regex_traits::transform_primary(_Fwd_iter __first, _Fwd_iter __last) const`**
Implement this function.

Member **`std::reverse_iterator::operator*() const`** Needs documentation! See
http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Member **`std::reverse_iterator::operator+(difference_type __n) const`** Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Member **`std::reverse_iterator::operator++()`** Needs documentation! See
http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Member **`std::reverse_iterator::operator++(int)`** Needs documentation! See
http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Member **`std::reverse_iterator::operator+=(difference_type __n)`** Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Member **`std::reverse_iterator::operator-(difference_type __n) const`** Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Member **`std::reverse_iterator::operator--(int)`** Needs documentation! See
http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Member **`std::reverse_iterator::operator--()`** Needs documentation! See
http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Member **`std::reverse_iterator::operator==(difference_type __n)`** Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

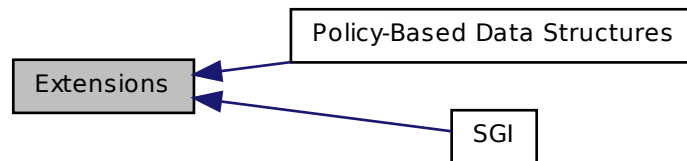
Member **`std::reverse_iterator::operator->() const`** Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Member **`std::reverse_iterator::operator[](difference_type __n) const`** Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

2 Module Documentation

2.1 Extensions

Collaboration diagram for Extensions:



Classes

- class `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>`
*Template class `__versa_string`.
 Data structure managing sequences of characters and character-like objects.*

Modules

- [SGI](#)
- [Policy-Based Data Structures](#)

2.1.1 Detailed Description

Components generally useful that are not part of any standard.

2.2 SGI

Collaboration diagram for SGI:



Classes

- class `__gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >`
An SGI extension .
- struct `__gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 >`
An SGI extension .
- struct `__gnu_cxx::constant_unary_fun< _Result, _Argument >`
An SGI extension .
- struct `__gnu_cxx::constant_void_fun< _Result >`
An SGI extension .
- class `__gnu_cxx::hash_map< _Key, _Tp, _HashFn, _EqualKey, _Alloc >`
- class `__gnu_cxx::hash_multimap< _Key, _Tp, _HashFn, _EqualKey, _Alloc >`
- class `__gnu_cxx::hash_multiset< _Value, _HashFcn, _EqualKey, _Alloc >`
- class `__gnu_cxx::hash_set< _Value, _HashFcn, _EqualKey, _Alloc >`
- struct `__gnu_cxx::project1st< _Arg1, _Arg2 >`
An SGI extension .
- struct `__gnu_cxx::project2nd< _Arg1, _Arg2 >`
An SGI extension .

- struct `__gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >`
- class `__gnu_cxx::rope< _CharT, _Alloc >`
- struct `__gnu_cxx::select1st< _Pair >`
An SGI extension .
- struct `__gnu_cxx::select2nd< _Pair >`
An SGI extension .
- class `__gnu_cxx::slist< _Tp, _Alloc >`
- class `__gnu_cxx::subtractive_rng`
- struct `__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >`
- class `__gnu_cxx::unary_compose< _Operation1, _Operation2 >`
An SGI extension .

Functions

- template<typename _Tp >
const _Tp & `__gnu_cxx::__median` (const _Tp &__a, const _Tp &__b, const _Tp &__c)
- template<typename _Tp, typename _Compare >
const _Tp & `__gnu_cxx::__median` (const _Tp &__a, const _Tp &__b, const _Tp &__c, _Compare __comp)
- size_t `std::bitset::Find_first` () const
- size_t `std::bitset::Find_next` (size_t __prev) const
- template<class _Operation1, class _Operation2 >
unary_compose< _Operation1, _Operation2 > `__gnu_cxx::compose1` (const _Operation1 &__fn1, const _Operation2 &__fn2)
- template<class _Operation1, class _Operation2, class _Operation3 >
binary_compose< _Operation1, _Operation2, _Operation3 > `__gnu_cxx::compose2` (const _Operation1 &__fn1, const _Operation2 &__fn2, const _Operation3 &__fn3)
- template<class _Result >
constant_void_fun< _Result > `__gnu_cxx::constant0` (const _Result &__val)
- template<class _Result >
constant_unary_fun< _Result, _Result > `__gnu_cxx::constant1` (const _Result &__val)
- template<class _Result >
constant_binary_fun< _Result, _Result, _Result > `__gnu_cxx::constant2` (const _Result &__val)
- template<typename _InputIterator, typename _Size, typename _OutputIterator >
pair< _InputIterator, _OutputIterator > `__gnu_cxx::copy_n` (_InputIterator __first, _Size __count, _OutputIterator __result)

- `template<typename _InputIterator, typename _Distance >`
`void __gnu_cxx::distance (_InputIterator __first, _InputIterator __last, _-`
`Distance &__n)`
- `template<class _Tp >`
`_Tp __gnu_cxx::identity_element (std::multiplies< _Tp >)`
- `template<class _Tp >`
`_Tp __gnu_cxx::identity_element (std::plus< _Tp >)`
- `template<typename _ForwardIter, typename _Tp >`
`void __gnu_cxx::iota (_ForwardIter __first, _ForwardIter __last, _Tp __value)`
- `template<typename _RandomAccessIterator >`
`bool __gnu_cxx::is_heap (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _StrictWeakOrdering >`
`bool __gnu_cxx::is_heap (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _StrictWeakOrdering __comp)`
- `template<typename _ForwardIterator >`
`bool __gnu_cxx::is_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _StrictWeakOrdering >`
`bool __gnu_cxx::is_sorted (_ForwardIterator __first, _ForwardIterator __last, _-`
`StrictWeakOrdering __comp)`
- `template<typename _InputIterator1, typename _InputIterator2 >`
`int __gnu_cxx::lexicographical_compare_3way (_InputIterator1 __first1, _-`
`InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _Tp, typename _Integer >`
`_Tp __gnu_cxx::power (_Tp __x, _Integer __n)`
- `template<typename _Tp, typename _Integer, typename _MonoidOperation >`
`_Tp __gnu_cxx::power (_Tp __x, _Integer __n, _MonoidOperation __monoid_-`
`op)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _-`
`RandomNumberGenerator >`
`_RandomAccessIterator __gnu_cxx::random_sample (_InputIterator _-`
`__first, _InputIterator __last, _RandomAccessIterator __out_first, _-`
`RandomAccessIterator __out_last, _RandomNumberGenerator &__rand)`
- `template<typename _InputIterator, typename _RandomAccessIterator >`
`_RandomAccessIterator __gnu_cxx::random_sample (_InputIterator _-`
`__first, _InputIterator __last, _RandomAccessIterator __out_first, _-`
`RandomAccessIterator __out_last)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance >`
`_OutputIterator __gnu_cxx::random_sample_n (_ForwardIterator __first, _-`
`ForwardIterator __last, _OutputIterator __out, const _Distance __n)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance, typename`
`_RandomNumberGenerator >`
`_OutputIterator __gnu_cxx::random_sample_n (_ForwardIterator __first, _-`
`ForwardIterator __last, _OutputIterator __out, const _Distance __n, _-`
`RandomNumberGenerator &__rand)`

- `template<typename _InputIter, typename _Size, typename _ForwardIter >`
`pair< _InputIter, _ForwardIter > __gnu_cxx::uninitialized_copy_n (_InputIter`
`__first, _Size __count, _ForwardIter __result)`
- `bitset< _Nb > & std::bitset::_Unchecked_set (size_t __pos)`
- `bitset< _Nb > & std::bitset::_Unchecked_set (size_t __pos, int __val)`
- `bitset< _Nb > & std::bitset::_Unchecked_reset (size_t __pos)`
- `bitset< _Nb > & std::bitset::_Unchecked_flip (size_t __pos)`
- `bool std::bitset::_Unchecked_test (size_t __pos) const`

2.2.1 Detailed Description

Because libstdc++ based its implementation of the STL subsections of the library on the SGI 3.3 implementation, we inherited their extensions as well.

They are additionally documented in the [online documentation](#), a copy of which is also shipped with the library source code (in `.../docs/html/documentation.html`). You can also read the documentation [on SGI's site](#), which is still running even though the code is not maintained.

NB that the following notes are pulled from various comments all over the place, so they may seem stilted.

The `identity_element` functions are not part of the C++ standard; SGI provided them as an extension. Its argument is an operation, and its return value is the identity element for that operation. It is overloaded for addition and multiplication, and you can overload it for your own nefarious operations.

As an extension to the binders, SGI provided composition functors and wrapper functions to aid in their creation. The `unary_compose` functor is constructed from two functions/functors, `f` and `g`. Calling `operator()` with a single argument `x` returns `f(g(x))`. The function `compose1` takes the two functions and constructs a `unary_compose` variable for you.

`binary_compose` is constructed from three functors, `f`, `g1`, and `g2`. Its `operator()` returns `f(g1(x),g2(x))`. The function takes `f`, `g1`, and `g2`, and constructs the `binary_compose` instance for you. For example, if `f` returns an `int`, then

```
int answer = (compose2(f,g1,g2))(x);
```

is equivalent to

```
int temp1 = g1(x);
int temp2 = g2(x);
int answer = f(temp1,temp2);
```

But the first form is more compact, and can be passed around as a functor to other algorithms.

As an extension, SGI provided a functor called `identity`. When a functor is required but no operations are desired, this can be used as a pass-through. Its `operator()` returns its argument unchanged.

`select1st` and `select2nd` are extensions provided by SGI. Their `operator()`s take a `std::pair` as an argument, and return either the first member or the second member, respectively. They can be used (especially with the composition functors) to *strip* data from a sequence before performing the remainder of an algorithm.

The `operator()` of the `project1st` functor takes two arbitrary arguments and returns the first one, while `project2nd` returns the second one. They are extensions provided by SGI.

These three functors are each constructed from a single arbitrary variable/value. Later, their `operator()`s completely ignore any arguments passed, and return the stored value.

- `constant_void_fun`'s `operator()` takes no arguments
- `constant_unary_fun`'s `operator()` takes one argument (ignored)
- `constant_binary_fun`'s `operator()` takes two arguments (ignored)

The helper creator functions `constant0`, `constant1`, and `constant2` each take a *result* argument and construct variables of the appropriate functor type.

2.2.2 Function Documentation

2.2.2.1 `template<typename _Tp> const _Tp& __gnu_cxx::__median (const _Tp & __a, const _Tp & __b, const _Tp & __c)`

Find the median of three values.

Parameters

- a* A value.
- b* A value.
- c* A value.

Returns

One of *a*, *b* or *c*.

If $\{1, m, n\}$ is some convolution of $\{a, b, c\}$ such that $1 \leq m \leq n$ then the value returned will be *m*. This is an SGI extension.

Definition at line 539 of file `ext/algorithm`.

2.2.2.2 `template<typename _Tp , typename _Compare > const _Tp&
 __gnu_cxx::__median (const _Tp & __a, const _Tp & __b, const _Tp
 & __c, _Compare __comp)`

Find the median of three values using a predicate for comparison.

Parameters

a A value.
b A value.
c A value.
comp A binary predicate.

Returns

One of *a*, *b* or *c*.

If { *l*, *m*, *n* } is some convolution of { *a*, *b*, *c* } such that `comp (l, m)` and `comp (m, n)` are both true then the value returned will be *m*. This is an SGI extension.

Definition at line 573 of file `ext/algorithm`.

2.2.2.3 `template<size_t _Nb> size_t std::bitset< _Nb >::_Find_first () const
 [inline, inherited]`

Finds the index of the first "on" bit.

Returns

The index of the first bit set, or `size()` if not found.

See also

`_Find_next`

Definition at line 1327 of file `bitset`.

2.2.2.4 `template<size_t _Nb> size_t std::bitset< _Nb >::_Find_next (size_t
 __prev) const [inline, inherited]`

Finds the index of the next "on" bit after *prev*.

Returns

The index of the next bit set, or [size\(\)](#) if not found.

Parameters

prev Where to start searching.

See also

[_Find_first](#)

Definition at line 1338 of file `bitset`.

2.2.2.5 `template<size_t _Nb> bitset<_Nb>& std::bitset< _Nb
>::_Unchecked_flip (size_t __pos) [inline, inherited]`

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 1010 of file `bitset`.

2.2.2.6 `template<size_t _Nb> bitset<_Nb>& std::bitset< _Nb
>::_Unchecked_reset (size_t __pos) [inline, inherited]`

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 1003 of file `bitset`.

2.2.2.7 `template<size_t _Nb> bitset<_Nb>& std::bitset< _Nb
>::_Unchecked_set (size_t __pos) [inline, inherited]`

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 986 of file `bitset`.

2.2.2.8 `template<size_t _Nb> bitset<_Nb>& std::bitset< _Nb
>::_Unchecked_set (size_t __pos, int __val) [inline,
inherited]`

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 993 of file `bitset`.

2.2.2.9 `template<size_t _Nb> bool std::bitset<_Nb>::_Unchecked_test (`
`size_t __pos) const [inline, inherited]`

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 1017 of file `bitset`.

2.2.2.10 `template<class _Operation1 , class _Operation2 >`
`unary_compose<_Operation1, _Operation2> __gnu_cxx::compose1`
`(const _Operation1 & __fn1, const _Operation2 & __fn2)`
`[inline]`

An [SGI extension](#) .

Definition at line 146 of file `ext/functional`.

2.2.2.11 `template<class _Operation1 , class _Operation2 , class _Operation3`
`> binary_compose<_Operation1, _Operation2, _Operation3>`
`__gnu_cxx::compose2 (const _Operation1 & __fn1, const`
`_Operation2 & __fn2, const _Operation3 & __fn3) [inline]`

An [SGI extension](#) .

Definition at line 173 of file `ext/functional`.

2.2.2.12 `template<class _Result > constant_void_fun<_Result>`
`__gnu_cxx::constant0 (const _Result & __val) [inline]`

An [SGI extension](#) .

Definition at line 327 of file `ext/functional`.

2.2.2.13 `template<class _Result > constant_unary_fun<_Result, _Result>`
`__gnu_cxx::constant1 (const _Result & __val) [inline]`

An [SGI extension](#) .

Definition at line 333 of file `ext/functional`.

2.2.2.14 `template<class _Result > constant_binary_fun<_Result, _Result, _Result> __gnu_cxx::constant2 (const _Result & __val)
[inline]`

An [SGI extension](#).

Definition at line 339 of file ext/functional.

2.2.2.15 `template<typename _InputIterator , typename _Size , typename _OutputIterator > pair<_InputIterator, _OutputIterator>
__gnu_cxx::copy_n (_InputIterator __first, _Size __count,
_OutputIterator __result) [inline]`

Copies the range [first,first+count) into [result,result+count).

Parameters

first An input iterator.

count The number of elements to copy.

result An output iterator.

Returns

A [std::pair](#) composed of first+count and result+count.

This is an SGI extension. This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Definition at line 121 of file ext/algorithm.

References `std::__iterator_category()`.

2.2.2.16 `template<typename _InputIterator , typename _Distance > void
__gnu_cxx::distance (_InputIterator __first, _InputIterator __last,
_Distance & __n) [inline]`

This is an SGI extension.

Todo

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Definition at line 105 of file ext/iterator.

References `std::__iterator_category()`.

Referenced by `__gnu_parallel::multiseq_partition()`, `__gnu_parallel::multiseq_selection()`, and `__gnu_cxx::random_sample_n()`.

2.2.2.17 `template<class _Tp > _Tp __gnu_cxx::identity_element (std::plus<_Tp >) [inline]`

An [SGI extension](#) .

Definition at line 88 of file ext/functional.

2.2.2.18 `template<class _Tp > _Tp __gnu_cxx::identity_element (std::multiplies<_Tp >) [inline]`

An [SGI extension](#) .

Definition at line 94 of file ext/functional.

2.2.2.19 `template<typename _ForwardIter , typename _Tp > void
__gnu_cxx::iota (_ForwardIter __first, _ForwardIter __last, _Tp
__value)`

This is an SGI extension.

Todo

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Definition at line 134 of file ext/numeric.

2.2.2.20 `template<typename _RandomAccessIterator , typename
_StrictWeakOrdering > bool __gnu_cxx::is_heap (
_RandomAccessIterator __first, _RandomAccessIterator __last,
_StrictWeakOrdering __comp) [inline]`

This is an SGI extension.

Todo

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Definition at line 455 of file ext/algorithm.

```
2.2.2.21  template<typename _RandomAccessIterator > bool
           __gnu_cxx::is_heap ( _RandomAccessIterator __first,
                               _RandomAccessIterator __last ) [inline]
```

This is an SGI extension.

Todo

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Definition at line 436 of file ext/algorithm.

```
2.2.2.22  template<typename _ForwardIterator > bool __gnu_cxx::is_sorted (
           _ForwardIterator __first, _ForwardIterator __last )
```

This is an SGI extension.

Todo

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Definition at line 480 of file ext/algorithm.

```
2.2.2.23  template<typename _ForwardIterator , typename
           _StrictWeakOrdering > bool __gnu_cxx::is_sorted (
           _ForwardIterator __first, _ForwardIterator __last,
           _StrictWeakOrdering __comp )
```

This is an SGI extension.

Todo

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Definition at line 505 of file ext/algorithm.

2.2.2.24 `template<typename _InputIterator1, typename _InputIterator2 >
 int __gnu_cxx::lexicographical_compare_3way (_InputIterator1
 __first1, _InputIterator1 __last1, _InputIterator2 __first2,
 _InputIterator2 __last2)`

memcmp on steroids.

Parameters

first1 An input iterator.

last1 An input iterator.

first2 An input iterator.

last2 An input iterator.

Returns

An int, as with memcmp.

The return value will be less than zero if the first range is *lexigraphically less than* the second, greater than zero if the second range is *lexigraphically less than* the first, and zero otherwise. This is an SGI extension.

Definition at line 202 of file ext/algorithm.

2.2.2.25 `template<typename _Tp, typename _Integer > _Tp
 __gnu_cxx::power (_Tp __x, _Integer __n) [inline]`

This is an SGI extension.

Todo

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Definition at line 123 of file ext/numeric.

2.2.2.26 `template<typename _Tp, typename _Integer, typename
 _MonoidOperation > _Tp __gnu_cxx::power (_Tp __x, _Integer
 __n, _MonoidOperation __monoid_op) [inline]`

This is an SGI extension.

Todo

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Definition at line 113 of file ext/numeric.

```
2.2.2.27 template<typename _InputIterator , typename
    _RandomAccessIterator , typename _RandomNumberGenerator >
    _RandomAccessIterator __gnu_cxx::random_sample ( _InputIterator
    __first, _InputIterator __last, _RandomAccessIterator __out_first,
    _RandomAccessIterator __out_last, _RandomNumberGenerator &
    __rand ) [inline]
```

This is an SGI extension.

Todo

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Definition at line 412 of file ext/algorithm.

```
2.2.2.28 template<typename _InputIterator , typename
    _RandomAccessIterator > _RandomAccessIterator
    __gnu_cxx::random_sample ( _InputIterator __first, _InputIterator
    __last, _RandomAccessIterator __out_first, _RandomAccessIterator
    __out_last ) [inline]
```

This is an SGI extension.

Todo

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Definition at line 389 of file ext/algorithm.

```
2.2.2.29 template<typename _ForwardIterator , typename _OutputIterator ,
    typename _Distance , typename _RandomNumberGenerator >
    _OutputIterator __gnu_cxx::random_sample_n ( _ForwardIterator
    __first, _ForwardIterator __last, _OutputIterator __out, const
    _Distance __n, _RandomNumberGenerator & __rand )
```

This is an SGI extension.

Todo

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Definition at line 302 of file ext/algorithm.

References `__gnu_cxx::distance()`, and `std::min()`.

```
2.2.2.30  template<typename _ForwardIterator , typename _OutputIterator ,
            typename _Distance > _OutputIterator __gnu_cxx::random_sample_n
            ( _ForwardIterator __first, _ForwardIterator __last,
              _OutputIterator __out, const _Distance __n )
```

This is an SGI extension.

Todo

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Definition at line 268 of file ext/algorithm.

References `__gnu_cxx::distance()`, and `std::min()`.

```
2.2.2.31  template<typename _InputIter , typename _Size ,
            typename _ForwardIter > pair<_InputIter, _ForwardIter>
            __gnu_cxx::uninitialized_copy_n ( _InputIter __first, _Size __count,
              _ForwardIter __result ) [inline]
```

Copies the range `[first,last)` into `result`.

Parameters

first An input iterator.

last An input iterator.

result An output iterator.

Returns

`result + (first - last)`

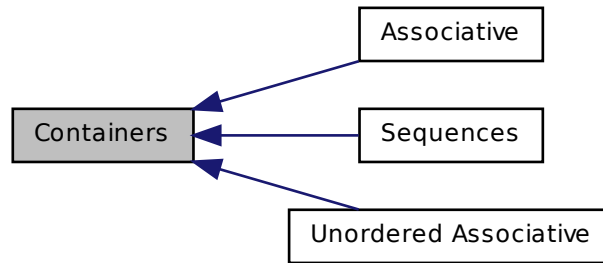
Like `copy()`, but does not require an initialized output range.

Definition at line 123 of file ext/memory.

References `std::__iterator_category()`.

2.3 Containers

Collaboration diagram for Containers:



Classes

- class `std::bitset<_Nb>`

The `bitset` class represents a fixed-size sequence of bits.

Modules

- `Sequences`
- `Associative`
- `Unordered Associative`

2.3.1 Detailed Description

Containers are collections of objects.

A container may hold any type which meets certain requirements, but the type of contained object is chosen at compile time, and all objects in a given container must be of the same type. (Polymorphism is possible by declaring a container of pointers to a base class and then populating it with pointers to instances of derived classes. Variant value types such as the `any` class from `Boost` can also be used.

All contained types must be `Assignable` and `CopyConstructible`. Specific containers may place additional requirements on the types of their contained objects.

Containers manage memory allocation and deallocation themselves when storing your objects. The objects are destroyed when the container is itself destroyed. Note that if you are storing pointers in a container, `delete` is *not* automatically called on the pointers before destroying them.

All containers must meet certain requirements, summarized in [tables](#).

The standard containers are further refined into [Sequences](#) and [Associative Containers](#). [Unordered Associative Containers](#).

2.4 Sequences

Collaboration diagram for Sequences:



Classes

- struct `std::array<_Tp, _Nm>`
A standard container for storing a fixed size sequence of elements.
- class `std::basic_string<_CharT, _Traits, _Alloc>`
Managing sequences of characters and character-like objects.
- class `std::deque<_Tp, _Alloc>`
A standard container using fixed-size memory allocation and constant-time manipulation of elements at either end.
- class `std::forward_list<_Tp, _Alloc>`
A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.
- class `std::list<_Tp, _Alloc>`
A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.

- class `std::priority_queue<_Tp, _Sequence, _Compare>`
A standard container automatically sorting its contents.
- class `std::queue<_Tp, _Sequence>`
A standard container giving FIFO behavior.
- class `std::stack<_Tp, _Sequence>`
A standard container giving FILO behavior.
- class `std::vector<_Tp, _Alloc>`
A standard container which offers fixed time access to individual elements in any order.
- class `std::vector<bool, _Alloc>`
A specialization of vector for booleans which offers fixed time access to individual elements in any order.

2.4.1 Detailed Description

Sequences arrange a collection of objects into a strictly linear order.

The differences between sequences are usually due to one or both of the following:

- memory management
- algorithmic complexity

As an example of the first case, `vector` is required to use a contiguous memory layout, while other sequences such as `deque` are not.

The prime reason for choosing one sequence over another should be based on the second category of differences, algorithmic complexity. For example, if you need to perform many inserts and removals from the middle of a sequence, `list` would be ideal. But if you need to perform constant-time access to random elements of the sequence, then `list` should not be used.

All sequences must meet certain requirements, summarized in [tables](#).

2.5 Associative

Collaboration diagram for Associative:



Classes

- class `std::map< _Key, _Tp, _Compare, _Alloc >`
A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.
- class `std::multimap< _Key, _Tp, _Compare, _Alloc >`
A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.
- class `std::multiset< _Key, _Compare, _Alloc >`
A standard container made up of elements, which can be retrieved in logarithmic time.
- class `std::set< _Key, _Compare, _Alloc >`
A standard container made up of unique keys, which can be retrieved in logarithmic time.

2.5.1 Detailed Description

Associative containers allow fast retrieval of data based on keys.

Each container type is parameterized on a `Key` type, and an ordering relation used to sort the elements of the container.

All associative containers must meet certain requirements, summarized in [tables](#).

2.6 Unordered Associative

Collaboration diagram for Unordered Associative:



Classes

- class `std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`
A standard container composed of unique keys (containing at most one of each key value) that associates values of another type with the keys.
- class `std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`
A standard container composed of equivalent keys (possibly containing multiple of each key value) that associates values of another type with the keys.
- class `std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`
A standard container composed of equivalent keys (possibly containing multiple of each key value) in which the elements' keys are the elements themselves.
- class `std::unordered_set< _Value, _Hash, _Pred, _Alloc >`
A standard container composed of unique keys (containing at most one of each key value) in which the elements' keys are the elements themselves.

2.6.1 Detailed Description

Unordered associative containers allow fast retrieval of data based on keys.

Each container type is parameterized on a `Key` type, a `Hash` type providing a hashing functor, and an ordering relation used to sort the elements of the container.

All unordered associative containers must meet certain requirements, summarized in [tables](#).

2.7 Diagnostics

Collaboration diagram for Diagnostics:



Modules

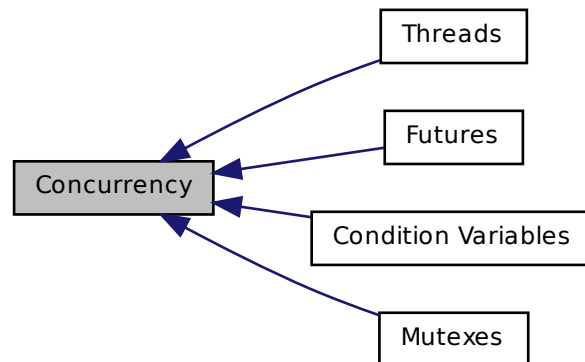
- [Exceptions](#)

2.7.1 Detailed Description

Components for error handling, reporting, and diagnostic operations.

2.8 Concurrency

Collaboration diagram for Concurrency:



Modules

- [Condition Variables](#)
- [Futures](#)
- [Mutexes](#)
- [Threads](#)

2.8.1 Detailed Description

Components for concurrent operations, including threads, mutexes, and condition variables.

2.9 Exceptions

Collaboration diagram for Exceptions:



Classes

- class `__cxxabiv1::__forced_unwind`
*Thrown as part of forced unwinding.
 A magic placeholder class that can be caught by reference to recognize forced unwinding.*
- struct `__gnu_cxx::forced_error`
Thrown by exception safety machinery.
- class `__gnu_cxx::recursive_init_error`
*Exception thrown by `__cxa_guard_acquire`.
 6.7[stmt.dcl]/4: If control re-enters the declaration (recursively) while the object is being initialized, the behavior is undefined.*
- class `std::__exception_ptr::exception_ptr`
An opaque pointer to an arbitrary exception.
- class `std::bad_alloc`
*Exception possibly thrown by `new`.
`bad_alloc` (or classes derived from it) is used to report allocation errors from the throwing forms of `new`.*
- class `std::bad_cast`
*Thrown during incorrect typecasting.
 If you attempt an invalid `dynamic_cast` expression, an instance of this class (or something derived from this class) is thrown.*
- class `std::bad_exception`
- class `std::bad_function_call`

Exception class thrown when class template function's operator() is called with an empty target.

- class `std::bad_typeid`
Thrown when a NULL pointer in a typeid expression is used.
- class `std::bad_weak_ptr`
Exception possibly thrown by `shared_ptr`.
- class `std::domain_error`
- class `std::exception`
Base class for all library exceptions.
- class `std::future_error`
Exception type thrown by futures.
- class `std::invalid_argument`
- class `std::ios_base::failure`
*These are thrown to indicate problems with io.
27.4.2.1.1 Class `ios_base::failure`.*
- class `std::length_error`
- class `std::logic_error`
One of two subclasses of exception.
- class `std::nested_exception`
Exception class with `exception_ptr` data member.
- class `std::out_of_range`
- class `std::overflow_error`
- class `std::range_error`
- class `std::regex_error`
*A regular expression exception class.
The regular expression library throws objects of this class on error.*
- class `std::runtime_error`
One of two subclasses of exception.
- class `std::system_error`
Thrown to indicate error code of underlying system.
- class `std::underflow_error`

Typedefs

- typedef void(* [std::terminate_handler](#))()
- typedef void(* [std::unexpected_handler](#))()

Functions

- template<typename _Ex >
const nested_exception * [std::__get_nested_exception](#) (const _Ex &__ex)
- template<typename _Ex >
void [std::__throw_with_nested](#) (_Ex &&, const nested_exception *__ex) __attribute__((__noreturn__))
- template<typename _Ex >
void [std::__throw_with_nested](#) (_Ex &&, ...) __attribute__((__noreturn__))
- void [__gnu_cxx::__verbose_terminate_handler](#) ()
- template<typename _Ex >
exception_ptr [std::copy_exception](#) (_Ex __ex) throw ()
- exception_ptr [std::current_exception](#) () throw ()
- template<typename _Ex >
exception_ptr [std::make_exception_ptr](#) (_Ex __ex) throw ()
- void [std::rethrow_exception](#) (exception_ptr) __attribute__((__noreturn__))
- void [std::rethrow_if_nested](#) (const nested_exception &__ex)
- template<typename _Ex >
void [std::rethrow_if_nested](#) (const _Ex &__ex)
- terminate_handler [std::set_terminate](#) (terminate_handler) throw ()
- unexpected_handler [std::set_unexpected](#) (unexpected_handler) throw ()
- void [std::terminate](#) () __attribute__((__noreturn__)) throw ()
- template<typename _Ex >
void [std::throw_with_nested](#) (_Ex __ex)
- bool [std::uncaught_exception](#) () __attribute__((__pure__)) throw ()
- void [std::unexpected](#) () __attribute__((__noreturn__))

2.9.1 Detailed Description

Classes and functions for reporting errors via exception classes.

2.9.2 Typedef Documentation

2.9.2.1 typedef void(* [std::terminate_handler](#))()

If you write a replacement terminate handler, it must be of this type.

Definition at line 88 of file exception.

2.9.2.2 typedef void(* std::unexpected_handler)()

If you write a replacement unexpected handler, it must be of this type.

Definition at line 91 of file exception.

2.9.3 Function Documentation

2.9.3.1 void __gnu_cxx::__verbose_terminate_handler ()

A replacement for the standard terminate_handler which prints more information about the terminating exception (if any) on stderr.

Call

```
std::set_terminate(__gnu_cxx::__verbose_terminate_handler)
```

to use. For more info, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt02ch06s02>.

In 3.4 and later, this is on by default.

2.9.3.2 template<typename _Ex > exception_ptr std::copy_exception (_Ex __ex) throw ()

Obtain an exception_ptr pointing to a copy of the supplied object.

Definition at line 162 of file exception_ptr.h.

References std::current_exception().

2.9.3.3 exception_ptr std::current_exception () throw ()

Obtain an exception_ptr to the currently handled exception. If there is none, or the currently handled exception is foreign, return the null value.

Referenced by std::copy_exception().

2.9.3.4 template<typename _Ex > exception_ptr std::make_exception_ptr (_Ex __ex) throw ()

Obtain an `exception_ptr` pointing to a copy of the supplied object.

Definition at line 181 of file `exception_ptr.h`.

2.9.3.5 `void std::rethrow_exception (exception_ptr)`

Throw the object pointed to by the `exception_ptr`.

Referenced by `std::__basic_future<_Res &>::M_get_result()`.

2.9.3.6 `void std::rethrow_if_nested (const nested_exception & __ex)` `[inline]`

Overload, See N2619.

Definition at line 156 of file `nested_exception.h`.

2.9.3.7 `template<typename _Ex> void std::rethrow_if_nested (const _Ex & __ex)` `[inline]`

If `__ex` is derived from [nested_exception](#), `__ex.rethrow_nested()`.

Definition at line 148 of file `nested_exception.h`.

2.9.3.8 `terminate_handler std::set_terminate (terminate_handler) throw ()`

Takes a new handler function as an argument, returns the old function.

2.9.3.9 `unexpected_handler std::set_unexpected (unexpected_handler)` `throw ()`

Takes a new handler function as an argument, returns the old function.

2.9.3.10 void std::terminate () throw ()

The runtime will call this function if exception handling must be abandoned for any reason. It can also be called by the user.

**2.9.3.11 template<typename _Ex> void std::throw_with_nested (_Ex __ex)
[inline]**

If __ex is derived from [nested_exception](#), __ex. // Else, an implementation-defined object derived from both.

Definition at line 138 of file nested_exception.h.

2.9.3.12 bool std::uncaught_exception () throw ()

[18.6.4]/1: 'Returns true after completing evaluation of a throw-expression until either completing initialization of the exception-declaration in the matching handler or entering [unexpected\(\)](#) due to the throw; or after entering [terminate\(\)](#) for any reason other than an explicit call to [terminate\(\)](#). [Note: This includes stack unwinding [15.2]. end note]'

2: 'When [uncaught_exception\(\)](#) is true, throwing an exception can result in a call of [terminate\(\)](#) (15.5.1).'

Referenced by std::basic_ostream<_CharT, _Traits>::sentry::~sentry().

2.9.3.13 void std::unexpected ()

The runtime will call this function if an exception is thrown which violates the function's exception specification.

2.10 Time

Collaboration diagram for Time:



Namespaces

- namespace [std::chrono](#)

2.10.1 Detailed Description

Classes and functions for time.

2.11 Complex Numbers

Collaboration diagram for Complex Numbers:



Classes

- struct [std::complex<_Tp>](#)

Functions

- constexpr **std::complex< float >::complex** (const complex< double > &)
- constexpr **std::complex< float >::complex** (const complex< long double > &)
- constexpr **std::complex< double >::complex** (const complex< long double > &)
- template<typename _Tp >
_Tp **std::__complex_abs** (const complex< _Tp > &__z)
- template<typename _Tp >
std::complex< _Tp > std::tr1::__complex_acos (const **std::complex< _Tp >**
&__z)
- template<typename _Tp >
std::complex< _Tp > std::tr1::__complex_acosh (const **std::complex< _Tp >**
&__z)
- template<typename _Tp >
_Tp **std::__complex_arg** (const complex< _Tp > &__z)
- template<typename _Tp >
std::complex< _Tp > std::tr1::__complex_asin (const **std::complex< _Tp >**
&__z)
- template<typename _Tp >
std::complex< _Tp > std::tr1::__complex_asinh (const **std::complex< _Tp >**
&__z)
- template<typename _Tp >
std::complex< _Tp > std::tr1::__complex_atan (const **std::complex< _Tp >**
&__z)
- template<typename _Tp >
std::complex< _Tp > std::tr1::__complex_atanh (const **std::complex< _Tp >**
&__z)
- template<typename _Tp >
complex< _Tp > **std::__complex_cos** (const complex< _Tp > &__z)
- template<typename _Tp >
complex< _Tp > **std::__complex_cosh** (const complex< _Tp > &__z)
- template<typename _Tp >
complex< _Tp > **std::__complex_exp** (const complex< _Tp > &__z)
- template<typename _Tp >
complex< _Tp > **std::__complex_log** (const complex< _Tp > &__z)
- template<typename _Tp >
complex< _Tp > **std::__complex_pow** (const complex< _Tp > &__x, const
complex< _Tp > &__y)
- template<typename _Tp >
complex< _Tp > **std::__complex_sin** (const complex< _Tp > &__z)
- template<typename _Tp >
complex< _Tp > **std::__complex_sinh** (const complex< _Tp > &__z)

- `template<typename _Tp >`
`complex< _Tp > std::__complex_sqrt (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`complex< _Tp > std::__complex_tan (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`complex< _Tp > std::__complex_tanh (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`_Tp std::abs (const complex< _Tp > &)`
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::acos (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::acosh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`_Tp std::arg (const complex< _Tp > &)`
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::asin (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::asinh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::atan (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::atanh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`complex< _Tp > std::conj (const complex< _Tp > &)`
- `template<typename _Tp >`
`std::complex< typename __gnu_cxx::__promote< _Tp >::__type >`
`std::tr1::conj (_Tp __x)`
- `template<typename _Tp >`
`complex< _Tp > std::cos (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::cosh (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::exp (const complex< _Tp > &)`
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::fabs (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`constexpr _Tp std::imag (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`complex< _Tp > std::log (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::log10 (const complex< _Tp > &)`
- `template<typename _Tp >`
`_Tp std::norm (const complex< _Tp > &)`
- `complex< _Tp > & std::complex::operator*= (const _Tp &)`

- `template<typename _Up >`
`complex< _Tp > & std::complex::operator*= (const complex< _Up > &)`
- `template<typename _Tp >`
`complex< _Tp > std::operator+ (const complex< _Tp > &__x)`
- `template<typename _Up >`
`complex< _Tp > & std::complex::operator+= (const complex< _Up > &)`
- `template<typename _Tp >`
`complex< _Tp > std::operator- (const complex< _Tp > &__x)`
- `template<typename _Up >`
`complex< _Tp > & std::complex::operator-= (const complex< _Up > &)`
- `template<typename _Up >`
`complex< _Tp > & std::complex::operator/= (const complex< _Up > &)`
- `complex< _Tp > & std::complex::operator/= (const _Tp &)`
- `template<typename _Tp, typename _CharT, class _Traits >`
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, const complex< _Tp > &__x)`
- `template<typename _Up >`
`complex< _Tp > & std::complex::operator= (const complex< _Up > &)`
- `complex< _Tp > & std::complex::operator= (const _Tp &)`
- `template<typename _Tp, typename _CharT, class _Traits >`
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, complex< _Tp > &__x)`
- `template<typename _Tp >`
`complex< _Tp > std::polar (const _Tp &, const _Tp &=0)`
- `template<typename _Tp, typename _Up >`
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >`
`std::tr1::polar (const _Tp &__rho, const _Up &__theta)`
- `template<typename _Tp >`
`complex< _Tp > std::pow (const complex< _Tp > &, const _Tp &)`
- `template<typename _Tp, typename _Up >`
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >`
`std::tr1::pow (const _Tp &__x, const std::complex< _Up > &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::pow (const _Tp &, const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::pow (const complex< _Tp > &, const complex< _Tp > &)`
- `template<typename _Tp, typename _Up >`
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >`
`std::tr1::pow (const std::complex< _Tp > &__x, const std::complex< _Up > &__y)`
- `template<typename _Tp, typename _Up >`
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >`
`std::tr1::pow (const std::complex< _Tp > &__x, const _Up &__y)`

- `template<typename _Tp >`
`constexpr _Tp std::real (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`complex< _Tp > std::sin (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::sinh (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::sqrt (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::tan (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::tanh (const complex< _Tp > &)`

- `template<typename _Tp >`
`complex< _Tp > std::operator+ (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator+ (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator+ (const _Tp &__x, const complex< _Tp > &__y)`

- `template<typename _Tp >`
`complex< _Tp > std::operator- (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator- (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator- (const _Tp &__x, const complex< _Tp > &__y)`

- `template<typename _Tp >`
`complex< _Tp > std::operator* (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator* (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator* (const _Tp &__x, const complex< _Tp > &__y)`

- `template<typename _Tp >`
`complex< _Tp > std::operator/ (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator/ (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator/ (const _Tp &__x, const complex< _Tp > &__y)`

- `template<typename _Tp >`
`constexpr bool std::operator== (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`constexpr bool std::operator== (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`constexpr bool std::operator== (const _Tp &__x, const complex< _Tp > &__y)`

- `template<typename _Tp >`
`constexpr bool std::operator!= (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`constexpr bool std::operator!= (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`constexpr bool std::operator!= (const _Tp &__x, const complex< _Tp > &__y)`

2.11.1 Detailed Description

Classes and functions for complex numbers.

2.11.2 Function Documentation

2.11.2.1 `template<typename _Tp > _Tp std::abs (const complex< _Tp > &__z) [inline]`

Return magnitude of z .

Definition at line 596 of file `complex`.

Referenced by `std::tr1::fabs()`, `std::fabs()`, `std::binomial_distribution< _IntType >::operator()()`, and `std::poisson_distribution< _IntType >::operator()()`.

2.11.2.2 `template<typename _Tp> std::complex<_Tp> std::tr1::acos (`
`const std::complex<_Tp> & __z) [inline]`

`acos(__z)` [8.1.2].

Definition at line 89 of file `tr1/complex`.

2.11.2.3 `template<typename _Tp> std::complex<_Tp> std::tr1::acosh (`
`const std::complex<_Tp> & __z) [inline]`

`acosh(__z)` [8.1.5].

Definition at line 208 of file `tr1/complex`.

2.11.2.4 `template<typename _Tp> _Tp std::arg (const complex<_Tp> &`
`__z) [inline]`

Return phase angle of z .

Definition at line 623 of file `complex`.

Referenced by `std::arg()`.

2.11.2.5 `template<typename _Tp> std::complex<_Tp> std::tr1::asin (`
`const std::complex<_Tp> & __z) [inline]`

`asin(__z)` [8.1.3].

Definition at line 125 of file `tr1/complex`.

2.11.2.6 `template<typename _Tp> std::complex<_Tp> std::tr1::asinh (`
`const std::complex<_Tp> & __z) [inline]`

`asinh(__z)` [8.1.6].

Definition at line 247 of file `tr1/complex`.

2.11.2.7 `template<typename _Tp> std::complex<_Tp> std::tr1::atan (`
`const std::complex<_Tp> & __z) [inline]`

`atan(__z)` [8.1.4].

Definition at line 169 of file `tr1/complex`.

2.11.2.8 `template<typename _Tp> std::complex<_Tp> std::tr1::atanh (`
`const std::complex<_Tp> & __z) [inline]`

`atanh(__z)` [8.1.7].

Definition at line 291 of file `tr1/complex`.

2.11.2.9 `template<typename _Tp> complex<_Tp> std::conj (const`
`complex<_Tp> & __z) [inline]`

Return complex conjugate of z .

Definition at line 368 of file `tr1/complex`.

References `std::conj()`.

Referenced by `std::conj()`.

2.11.2.10 `template<typename _Tp> complex<_Tp> std::cos (const`
`complex<_Tp> & __z) [inline]`

Return complex cosine of z .

Definition at line 701 of file `complex`.

Referenced by `std::polar()`.

2.11.2.11 `template<typename _Tp> complex<_Tp> std::cosh (const`
`complex<_Tp> & __z) [inline]`

Return complex hyperbolic cosine of z .

Definition at line 731 of file `complex`.

2.11.2.12 `template<typename _Tp> complex<_Tp> std::exp (const
complex<_Tp> & __z) [inline]`

Return complex base e exponential of z .

Definition at line 757 of file `complex`.

2.11.2.13 `template<typename _Tp> std::complex<_Tp> std::tr1::fabs (const
std::complex<_Tp> & __z) [inline]`

`fabs(__z)` [8.1.8].

Definition at line 300 of file `tr1/complex`.

References `std::abs()`.

2.11.2.14 `template<typename _Tp> complex<_Tp> std::log (const
complex<_Tp> & __z) [inline]`

Return complex natural logarithm of z .

Definition at line 784 of file `complex`.

Referenced by `std::generate_canonical()`, `std::log10()`, `std::gamma_distribution<_RealType>::operator()`, `std::normal_distribution<_RealType>::operator()`, `std::binomial_distribution<_IntType>::operator()`, `std::poisson_distribution<_IntType>::operator()`, and `std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::operator()`.

2.11.2.15 `template<typename _Tp> complex<_Tp> std::log10 (const
complex<_Tp> & __z) [inline]`

Return complex base 10 logarithm of z .

Definition at line 789 of file `complex`.

References `std::log()`.

2.11.2.16 `template<typename _Tp > _Tp std::norm (const complex< _Tp > & __z) [inline]`

Return z magnitude squared.

Definition at line 656 of file complex.

Referenced by `std::complex< _Tp >::operator/=()`.

2.11.2.17 `template<typename _Tp > constexpr bool std::operator!= (const complex< _Tp > & __x, const complex< _Tp > & __y) [inline]`

Return false if x is equal to y .

Definition at line 471 of file complex.

2.11.2.18 `template<typename _Tp > constexpr bool std::operator!= (const complex< _Tp > & __x, const _Tp & __y) [inline]`

Return false if x is equal to y .

Definition at line 476 of file complex.

2.11.2.19 `template<typename _Tp > constexpr bool std::operator!= (const _Tp & __x, const complex< _Tp > & __y) [inline]`

Return false if x is equal to y .

Definition at line 481 of file complex.

2.11.2.20 `template<typename _Tp > complex<_Tp> std::operator* (const complex< _Tp > & __x, const complex< _Tp > & __y) [inline]`

Return new complex value x times y .

Definition at line 381 of file complex.

2.11.2.21 `template<typename _Tp> complex<_Tp> std::operator*(const
complex<_Tp> & __x, const _Tp & __y) [inline]`

Return new complex value x times y .

Definition at line 390 of file complex.

2.11.2.22 `template<typename _Tp> complex<_Tp> std::operator*(const
_Tp & __x, const complex<_Tp> & __y) [inline]`

Return new complex value x times y .

Definition at line 399 of file complex.

2.11.2.23 `template<typename _Tp> template<typename _Up> complex<
_Tp> & std::complex<_Tp>::operator*=(const complex<_Up>
& __z) [inherited]`

Multiply this complex number by z .

Definition at line 294 of file complex.

2.11.2.24 `template<typename _Tp> complex<_Tp> & std::complex<_Tp>
>::operator*=(const _Tp & __t) [inherited]`

Multiply this complex number by t .

Definition at line 240 of file complex.

2.11.2.25 `template<typename _Tp> complex<_Tp> std::operator+(
const complex<_Tp> & __x, const complex<_Tp> & __y)
[inline]`

Return new complex value x plus y .

Definition at line 321 of file complex.

2.11.2.26 `template<typename _Tp> complex<_Tp> std::operator+ (const
_Tp & __x, const complex<_Tp> & __y) [inline]`

Return new complex value x plus y .

Definition at line 339 of file complex.

2.11.2.27 `template<typename _Tp> complex<_Tp> std::operator+ (const
complex<_Tp> & __x, const _Tp & __y) [inline]`

Return new complex value x plus y .

Definition at line 330 of file complex.

2.11.2.28 `template<typename _Tp> complex<_Tp> std::operator+ (const
complex<_Tp> & __x) [inline]`

Return x .

Definition at line 440 of file complex.

2.11.2.29 `template<typename _Tp> template<typename _Up> complex<
_Tp> & std::complex<_Tp>::operator+=(const complex<_Up>
& __z) [inherited]`

Add z to this complex number.

Definition at line 271 of file complex.

2.11.2.30 `template<typename _Tp> complex<_Tp> std::operator- (const
complex<_Tp> & __x) [inline]`

Return complex negation of x .

Definition at line 446 of file complex.

2.11.2.31 `template<typename _Tp> complex<_Tp> std::operator- (`
`const complex<_Tp> & __x, const complex<_Tp> & __y)`
`[inline]`

Return new complex value x minus y .

Definition at line 351 of file complex.

2.11.2.32 `template<typename _Tp> complex<_Tp> std::operator- (const`
`complex<_Tp> & __x, const _Tp & __y) [inline]`

Return new complex value x minus y .

Definition at line 360 of file complex.

2.11.2.33 `template<typename _Tp> complex<_Tp> std::operator- (const`
`_Tp & __x, const complex<_Tp> & __y) [inline]`

Return new complex value x minus y .

Definition at line 369 of file complex.

2.11.2.34 `template<typename _Tp> template<typename _Up> complex<`
`_Tp> & std::complex<_Tp>::operator-= (const complex<_Up>`
`& __z) [inherited]`

Subtract z from this complex number.

Definition at line 282 of file complex.

2.11.2.35 `template<typename _Tp> complex<_Tp> std::operator/ (const`
`_Tp & __x, const complex<_Tp> & __y) [inline]`

Return new complex value x divided by y .

Definition at line 429 of file complex.

2.11.2.36 `template<typename _Tp > complex<_Tp> std::operator/ (`
`const complex<_Tp > & __x, const complex<_Tp > & __y)`
`[inline]`

Return new complex value x divided by y .

Definition at line 411 of file complex.

2.11.2.37 `template<typename _Tp > complex<_Tp> std::operator/ (const`
`complex<_Tp > & __x, const _Tp & __y) [inline]`

Return new complex value x divided by y .

Definition at line 420 of file complex.

2.11.2.38 `template<typename _Tp > complex<_Tp > & std::complex<_Tp`
`>::operator/=(const _Tp & __t) [inherited]`

Divide this complex number by t .

Definition at line 250 of file complex.

2.11.2.39 `template<typename _Tp > template<typename _Up > complex<`
`_Tp > & std::complex<_Tp >::operator/=(const complex<_Up >`
`& __z) [inherited]`

Divide this complex number by z .

Definition at line 307 of file complex.

References `std::norm()`.

2.11.2.40 `template<typename _Tp , typename _CharT , class _Traits`
`> basic_ostream<_CharT, _Traits> & std::operator<< (`
`basic_ostream<_CharT, _Traits > & __os, const complex<_Tp >`
`& __x)`

Insertion operator for complex values.

Definition at line 521 of file complex.

References `std::ios_base::flags()`, `std::basic_ios< _CharT, _Traits >::imbue()`, `std::ios_base::precision()`, and `std::basic_ostringstream< _CharT, _Traits, _Alloc >::str()`.

2.11.2.41 `template<typename _Tp> complex< _Tp > & std::complex< _Tp >::operator=(const _Tp & __t) [inherited]`

Assign this complex number to scalar t .

Definition at line 230 of file complex.

2.11.2.42 `template<typename _Tp> template<typename _Up> complex< _Tp > & std::complex< _Tp >::operator=(const complex< _Up > & __z) [inherited]`

Assign this complex number to complex z .

Definition at line 260 of file complex.

2.11.2.43 `template<typename _Tp> constexpr bool std::operator==(const complex< _Tp > & __x, const _Tp & __y) [inline]`

Return true if x is equal to y .

Definition at line 458 of file complex.

2.11.2.44 `template<typename _Tp> constexpr bool std::operator==(const complex< _Tp > & __x, const complex< _Tp > & __y) [inline]`

Return true if x is equal to y .

Definition at line 453 of file complex.

2.11.2.45 `template<typename _Tp > constexpr bool std::operator==(const
_Tp & __x, const complex<_Tp> & __y) [inline]`

Return true if x is equal to y .

Definition at line 463 of file `complex`.

2.11.2.46 `template<typename _Tp , typename _CharT , class _Traits
> basic_istream<_CharT, _Traits>& std::operator>> (
basic_istream<_CharT, _Traits> & __is, complex<_Tp> & __x)`

Extraction operator for complex values.

Definition at line 488 of file `complex`.

References `std::ios_base::failbit`, `std::basic_istream<_CharT, _Traits>::putback()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

2.11.2.47 `template<typename _Tp > complex<_Tp> std::polar (const _Tp
& __rho, const _Tp & __theta = 0) [inline]`

Return complex with magnitude ρ and angle θ .

Definition at line 664 of file `complex`.

References `std::cos()`, and `std::sin()`.

2.11.2.48 `template<typename _Tp > complex<_Tp> std::pow (const
complex<_Tp> & __x, const _Tp & __y)`

Return x to the y 'th power.

Definition at line 392 of file `tr1/complex`.

References `std::pow()`.

Referenced by `std::gamma_distribution<_RealType>::operator()()`, `std::pow()`, and `std::tr1::pow()`.

2.11.2.49 `template<typename _Tp , typename _Up > std::complex<typename
__gnu_cxx::__promote_2<_Tp, _Up>::__type> std::tr1::pow (
const std::complex< _Tp > & __x, const _Up & __y) [inline]`

Additional overloads [8.1.9].

Definition at line 341 of file tr1/complex.

References std::pow().

2.11.2.50 `template<typename _Tp > complex< _Tp > std::pow (const
complex< _Tp > & __x, const complex< _Tp > & __y)
[inline]`

Return x to the y 'th power.

Definition at line 402 of file tr1/complex.

References std::pow().

2.11.2.51 `template<typename _Tp > complex< _Tp > std::pow (const _Tp &
__x, const complex< _Tp > & __y) [inline]`

Return x to the y 'th power.

Definition at line 397 of file tr1/complex.

References std::pow().

2.11.2.52 `template<typename _Tp > complex< _Tp > std::sin (const
complex< _Tp > & __z) [inline]`

Return complex sine of z .

Definition at line 819 of file complex.

Referenced by std::polar().

2.11.2.53 `template<typename _Tp> complex<_Tp> std::sinh (const
complex<_Tp> & __z) [inline]`

Return complex hyperbolic sine of z .

Definition at line 849 of file `complex`.

2.11.2.54 `template<typename _Tp> complex<_Tp> std::sqrt (const
complex<_Tp> & __z) [inline]`

Return complex square root of z .

Definition at line 893 of file `complex`.

Referenced by `std::normal_distribution<_RealType>::operator()()`, and
`std::student_t_distribution<_RealType>::operator()()`.

2.11.2.55 `template<typename _Tp> complex<_Tp> std::tan (const
complex<_Tp> & __z) [inline]`

Return complex tangent of z .

Definition at line 920 of file `complex`.

2.11.2.56 `template<typename _Tp> complex<_Tp> std::tanh (const
complex<_Tp> & __z) [inline]`

Return complex hyperbolic tangent of z .

Definition at line 948 of file `complex`.

2.12 Condition Variables

Collaboration diagram for Condition Variables:



Classes

- class `std::condition_variable`
condition_variable
- class `std::condition_variable_any`
condition_variable_any

Enumerations

- enum `std::cv_status` { `no_timeout`, `timeout` }

2.12.1 Detailed Description

Classes for `condition_variable` support.

2.12.2 Enumeration Type Documentation

2.12.2.1 enum `std::cv_status`

`cv_status`

Definition at line 56 of file `condition_variable`.

2.13 Futures

Collaboration diagram for Futures:



Classes

- class `std::__basic_future< _Res >`
Common implementation for future and `shared_future`.
- struct `std::__future_base`
Base class and enclosing scope.
- struct `std::__future_base::_Result< _Res & >`
Partial specialization for reference types.
- struct `std::__future_base::_Result< void >`
Explicit specialization for void.
- class `std::future< _Res >`
Primary template for future.
- class `std::future< _Res & >`
Partial specialization for `future<R&>`
- class `std::future< void >`
Explicit specialization for `future<void>`
- class `std::future_error`
Exception type thrown by futures.
- struct `std::is_error_code_enum< future_errc >`
Specialization.

- class `std::packaged_task<_Res(_ArgTypes...)>`
packaged_task
- class `std::promise<_Res>`
Primary template for promise.
- class `std::promise<_Res &>`
Partial specialization for promise<R&>
- class `std::promise<void>`
Explicit specialization for promise<void>
- class `std::shared_future<_Res>`
Primary template for shared_future.
- class `std::shared_future<_Res &>`
Partial specialization for shared_future<R&>
- class `std::shared_future<void>`
Explicit specialization for shared_future<void>

Enumerations

- enum `std::future_errc` { `broken_promise`, `future_already_retrieved`, `promise_already_satisfied`, `no_state` }
- enum `std::future_status` { `ready`, `timeout`, `deferred` }
- enum `std::launch` { `any`, `async`, `sync` }

Functions

- `std::__basic_future::__basic_future` (const shared_future<_Res> &)
- `std::__basic_future::__basic_future` (shared_future<_Res> &&)
- `std::__basic_future::__basic_future` (future<_Res> &&)
- static _Setter<void, void> `std::__future_base::State::__setter` (promise<void> * __prom)
- template<typename _Fn, typename... _Args>
future<typename result_of<_Fn(_Args...)>::type> `std::async` (launch __policy, _Fn && __fn, _Args &&... __args)
- template<typename _Fn, typename... _Args>
enable_if<!is_same<typename decay<_Fn>::type, launch>::value, future<decltype(std::declval<_Fn>)(std::declval<_Args>)...)>::type> `std::async` (_Fn && __fn, _Args &&... __args)

- `const error_category & std::future_category ()`
- `error_code std::make_error_code (future_errc __errc)`
- `error_condition std::make_error_condition (future_errc __errc)`
- `void std::promise< void >::set_value ()`
- `template<typename _Res >
void std::swap (promise< _Res > &__x, promise< _Res > &__y)`
- `template<typename _Res, typename... _ArgTypes>
void std::swap (packaged_task< _Res(_ArgTypes...)> &__x, packaged_task< _Res(_ArgTypes...)> &__y)`

2.13.1 Detailed Description

Classes for futures support.

2.13.2 Enumeration Type Documentation

2.13.2.1 `enum std::future_errc`

Error code for futures.

Definition at line 61 of file future.

2.13.2.2 `enum std::future_status`

Status code for futures.

Definition at line 134 of file future.

2.13.2.3 `enum std::launch`

Launch code for futures.

Definition at line 126 of file future.

2.13.3 Function Documentation

2.13.3.1 `template<typename _Fn , typename... _Args> future< typename
result_of< _Fn(_Args...)>::type > std::async (launch __policy, _Fn
&& __fn, _Args &&... __args)`

async

Definition at line 1345 of file future.

2.13.3.2 `template<typename _Fn , typename... _Args> enable_if<!is_same<
typename decay< _Fn >::type, launch >::value, future<
decltype(std::declval< _Fn >)(std::declval< _Args >)...)> >::type
std::async (_Fn && __fn, _Args &&... __args) [inline]`

async, potential overload

Definition at line 1370 of file future.

2.13.3.3 `const error_category& std::future_category ()`

Points to a statically-allocated object derived from [error_category](#).

Referenced by `std::make_error_code()`, and `std::make_error_condition()`.

2.13.3.4 `error_code std::make_error_code (future_errc __errc) [inline]`

Overload for `make_error_code`.

Definition at line 79 of file future.

References `std::future_category()`.

2.13.3.5 `error_condition std::make_error_condition (future_errc __errc)
[inline]`

Overload for `make_error_condition`.

Definition at line 84 of file future.

References `std::future_category()`.

2.13.3.6 `template<typename _Res , typename... _ArgTypes> void std::swap
(packaged_task< _Res(_ArgTypes...)> & __x, packaged_task<
_Res(_ArgTypes...)> & __y) [inline]`

swap

Definition at line 1279 of file future.

2.14 I/O

Classes

- class `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`
*Provides a layer of compatibility for C/POSIX.
This GNU extension provides extensions for working with standard C FILE*'s and POSIX file descriptors. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.*
- class `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`
*Provides a layer of compatibility for C.
This GNU extension provides extensions for working with standard C FILE*'s. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.*
- class `std::basic_filebuf< _CharT, _Traits >`
*The actual work of input and output (for files).
This class associates both its input and output sequence with an external disk file, and maintains a joint file position for both sequences. Many of its semantics are described in terms of similar behavior in the Standard C Library's FILE streams.*
- class `std::basic_fstream< _CharT, _Traits >`
*Controlling input and output for files.
This class supports reading from and writing to named files, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as *sb*.*
- class `std::basic_ifstream< _CharT, _Traits >`
Controlling input for files.

This class supports reading from named files, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.

- class `std::basic_ios< _CharT, _Traits >`
*Virtual base class for all stream classes.
 Most of the member functions called dispatched on stream objects (e.g., `std::cout.foo(bar);`) are consolidated in this class.*
- class `std::basic_iostream< _CharT, _Traits >`
*Merging istream and ostream capabilities.
 This class multiply inherits from the input and output stream classes simply to provide a single interface.*
- class `std::basic_istream< _CharT, _Traits >`
*Controlling input.
 This is the base class for all input streams. It provides text formatting of all builtin types, and communicates with any class derived from `basic_streambuf` to do the actual input.*
- class `std::basic_istreamstream< _CharT, _Traits, _Alloc >`
*Controlling input for `std::string`.
 This class supports reading from objects of type `std::basic_string`, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.*
- class `std::basic_ofstream< _CharT, _Traits >`
*Controlling output for files.
 This class supports reading from named files, using the inherited functions from `std::basic_ostream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.*
- class `std::basic_ostream< _CharT, _Traits >`
*Controlling output.
 This is the base class for all output streams. It provides text formatting of all builtin types, and communicates with any class derived from `basic_streambuf` to do the actual output.*
- class `std::basic_ostreamstream< _CharT, _Traits, _Alloc >`
*Controlling output for `std::string`.
 This class supports writing to objects of type `std::basic_string`, using the inherited functions from `std::basic_ostream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.*
- class `std::basic_streambuf< _CharT, _Traits >`

The actual work of input and output (interface).

This is a base class. Derived stream buffers each control a pair of character sequences: one for input, and one for output.

- class `std::basic_stringbuf< _CharT, _Traits, _Alloc >`

The actual work of input and output (for `std::string`).

This class associates either or both of its input and output sequences with a sequence of characters, which can be initialized from, or made available as, a `std::basic_string`. (Paraphrased from [27.7.1]/1.).

- class `std::basic_stringstream< _CharT, _Traits, _Alloc >`

Controlling input and output for `std::string`.

This class supports reading from and writing to objects of type `std::basic_string`, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

- class `std::ios_base`

The base of the I/O class hierarchy.

This class defines everything that can be defined about I/O that does not depend on the type of characters being input or output. Most people will only see `ios_base` when they need to specify the full name of the various I/O flags (e.g., the openmodes).

Typedefs

- `typedef basic_filebuf< char > std::filebuf`
- `typedef basic_fstream< char > std::fstream`
- `typedef basic_ifstream< char > std::ifstream`
- `typedef basic_ios< char > std::ios`
- `typedef basic_istream< char > std::istream`
- `typedef basic_istream< char > std::istream`
- `typedef basic_istreamstream< char > std::istreamstream`
- `typedef basic_ofstream< char > std::ofstream`
- `typedef basic_ostream< char > std::ostream`
- `typedef basic_ostreamstream< char > std::ostreamstream`
- `typedef basic_streambuf< char > std::streambuf`
- `typedef basic_stringbuf< char > std::stringbuf`
- `typedef basic_stringstream< char > std::stringstream`
- `typedef basic_filebuf< wchar_t > std::wfilebuf`
- `typedef basic_fstream< wchar_t > std::wfstream`
- `typedef basic_ifstream< wchar_t > std::wifstream`
- `typedef basic_ios< wchar_t > std::wios`
- `typedef basic_istream< wchar_t > std::wistream`
- `typedef basic_istream< wchar_t > std::wistream`

- `typedef basic_istream< wchar_t > std::wistream`
- `typedef basic_ofstream< wchar_t > std::wofstream`
- `typedef basic_ostream< wchar_t > std::wostream`
- `typedef basic_ostringstream< wchar_t > std::wostringstream`
- `typedef basic_streambuf< wchar_t > std::wstreambuf`
- `typedef basic_stringbuf< wchar_t > std::wstringbuf`
- `typedef basic_stringstream< wchar_t > std::wstringstream`

2.14.1 Detailed Description

Nearly all of the I/O classes are parameterized on the type of characters they read and write. (The major exception is `ios_base` at the top of the hierarchy.) This is a change from pre-Standard streams, which were not templates.

For ease of use and compatibility, all of the `basic_*` I/O-related classes are given typedef names for both of the builtin character widths (wide and narrow). The typedefs are the same as the pre-Standard names, for example:

```
typedef basic_ifstream<char> ifstream;
```

Because properly forward-declaring these classes can be difficult, you should not do it yourself. Instead, include the `<iosfwd>` header, which contains only declarations of all the I/O classes as well as the typedefs. Trying to forward-declare the typedefs themselves (e.g., `class ostream;`) is not valid ISO C++.

For more specific declarations, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch>

2.14.2 Typedef Documentation

2.14.2.1 `typedef basic_filebuf<char> std::filebuf`

Class for `char` file buffers.

Definition at line 156 of file `iosfwd`.

2.14.2.2 `typedef basic_fstream<char> std::fstream`

Class for `char` mixed input and output file streams.

Definition at line 165 of file `iosfwd`.

2.14.2.3 typedef basic_ifstream<char> std::ifstream

Class for `char` input file streams.

Definition at line 159 of file `iosfwd`.

2.14.2.4 typedef basic_ios<char> std::ios

Base class for `char` streams.

Definition at line 125 of file `iosfwd`.

2.14.2.5 typedef basic_iostream<char> std::iostream

Base class for `char` mixed input and output streams.

Definition at line 141 of file `iosfwd`.

2.14.2.6 typedef basic_istream<char> std::istream

Base class for `char` input streams.

Definition at line 135 of file `iosfwd`.

2.14.2.7 typedef basic_istream<char> std::istream

Class for `char` input memory streams.

Definition at line 147 of file `iosfwd`.

2.14.2.8 typedef basic_ofstream<char> std::ofstream

Class for `char` output file streams.

Definition at line 162 of file `iosfwd`.

2.14.2.9 typedef basic_ostream<char> std::ostream

Base class for `char` output streams.

Definition at line 138 of file `iosfwd`.

2.14.2.10 typedef basic_ostringstream<char> std::ostringstream

Class for `char` output memory streams.

Definition at line 150 of file `iosfwd`.

2.14.2.11 typedef basic_streambuf<char> std::streambuf

Base class for `char` buffers.

Definition at line 132 of file `iosfwd`.

2.14.2.12 typedef basic_stringbuf<char> std::stringbuf

Class for `char` memory buffers.

Definition at line 144 of file `iosfwd`.

2.14.2.13 typedef basic_stringstream<char> std::stringstream

Class for `char` mixed input and output memory streams.

Definition at line 153 of file `iosfwd`.

2.14.2.14 typedef basic_filebuf<wchar_t> std::wfilebuf

Class for `wchar_t` file buffers.

Definition at line 196 of file `iosfwd`.

2.14.2.15 typedef basic_fstream<wchar_t> std::wfstream

Class for `wchar_t` mixed input and output file streams.

Definition at line 205 of file `iosfwd`.

2.14.2.16 typedef basic_ifstream<wchar_t> std::wifstream

Class for `wchar_t` input file streams.

Definition at line 199 of file `iosfwd`.

2.14.2.17 typedef basic_ios<wchar_t> std::wios

Base class for `wchar_t` streams.

Definition at line 169 of file `iosfwd`.

2.14.2.18 typedef basic_iostream<wchar_t> std::wiostream

Base class for `wchar_t` mixed input and output streams.

Definition at line 181 of file `iosfwd`.

2.14.2.19 typedef basic_istream<wchar_t> std::wistream

Base class for `wchar_t` input streams.

Definition at line 175 of file `iosfwd`.

2.14.2.20 typedef basic_istringstream<wchar_t> std::wistringstream

Class for `wchar_t` input memory streams.

Definition at line 187 of file `iosfwd`.

2.14.2.21 typedef basic_ofstream<wchar_t> std::wofstream

Class for `wchar_t` output file streams.

Definition at line 202 of file `iosfwd`.

2.14.2.22 typedef basic_ostream<wchar_t> std::wostream

Base class for `wchar_t` output streams.

Definition at line 178 of file `iosfwd`.

2.14.2.23 typedef basic_ostringstream<wchar_t> std::wostringstream

Class for `wchar_t` output memory streams.

Definition at line 190 of file `iosfwd`.

2.14.2.24 typedef basic_streambuf<wchar_t> std::wstreambuf

Base class for `wchar_t` buffers.

Definition at line 172 of file `iosfwd`.

2.14.2.25 typedef basic_stringbuf<wchar_t> std::wstringbuf

Class for `wchar_t` memory buffers.

Definition at line 184 of file `iosfwd`.

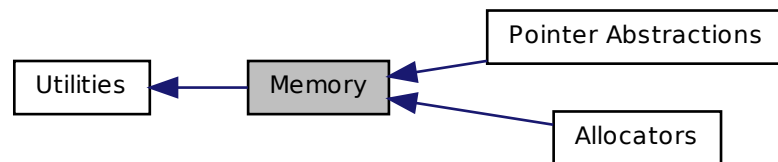
2.14.2.26 typedef basic_stringstream<wchar_t> std::wstringstream

Class for `wchar_t` mixed input and output memory streams.

Definition at line 193 of file `iosfwd`.

2.15 Memory

Collaboration diagram for Memory:



Modules

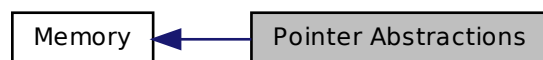
- [Pointer Abstractions](#)
- [Allocators](#)

2.15.1 Detailed Description

Components for memory allocation, deallocation, and management.

2.16 Pointer Abstractions

Collaboration diagram for Pointer Abstractions:



Classes

- struct `std::default_delete< _Tp >`
Primary template, `default_delete`.
- struct `std::default_delete< _Tp[]>`
Specialization, `default_delete`.
- class `std::enable_shared_from_this< _Tp >`
Base class allowing use of member function `shared_from_this`.
- struct `std::hash< shared_ptr< _Tp > >`
`std::hash` specialization for `shared_ptr`.
- struct `std::hash< unique_ptr< _Tp, _Dp > >`
`std::hash` specialization for `unique_ptr`.
- struct `std::owner_less< shared_ptr< _Tp > >`
Partial specialization of `owner_less` for `shared_ptr`.
- struct `std::owner_less< weak_ptr< _Tp > >`
Partial specialization of `owner_less` for `weak_ptr`.
- class `std::shared_ptr< _Tp >`
A smart pointer with reference-counted copy semantics.
- class `std::unique_ptr< _Tp, _Dp >`
20.7.12.2 `unique_ptr` for single objects.
- class `std::unique_ptr< _Tp[], _Dp >`
20.7.12.3 `unique_ptr` for array objects with a runtime length
- class `std::weak_ptr< _Tp >`
A smart pointer with weak semantics.

Functions

- template<typename _Tp, typename _Alloc, typename... _Args>
shared_ptr< _Tp > `std::allocate_shared` (const _Alloc &__a, _Args &&...__args)

- `template<typename _Tp, typename _Tp1 >`
`shared_ptr< _Tp > std::const_pointer_cast (const shared_ptr< _Tp1 > &__r)`
- `template<typename _Tp, typename _Tp1 >`
`shared_ptr< _Tp > std::dynamic_pointer_cast (const shared_ptr< _Tp1 > &__r)`
- `template<typename _Del, typename _Tp, _Lock_policy _Lp>`
`_Del * std::get_deleter (const __shared_ptr< _Tp, _Lp > &__p)`
- `template<typename _Tp, typename... _Args>`
`shared_ptr< _Tp > std::make_shared (_Args &&... __args)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator!= (nullptr_t, const unique_ptr< _Tp, _Dp > &__y)`
- `template<typename _Tp >`
`bool std::operator!= (nullptr_t, const shared_ptr< _Tp > &__b)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool std::operator!= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator!= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp1, typename _Tp2 >`
`bool std::operator!= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b)`
- `template<typename _Tp >`
`bool std::operator!= (const shared_ptr< _Tp > &__a, nullptr_t)`
- `template<typename _Tp1, typename _Tp2 >`
`bool std::operator< (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool std::operator< (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp>`
`std::basic_ostream< _Ch, _Tr > & std::operator<< (std::basic_ostream< _Ch, _Tr > &__os, const __shared_ptr< _Tp, _Lp > &__p)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool std::operator<= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp1, typename _Tp2 >`
`bool std::operator== (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b)`
- `template<typename _Tp >`
`bool std::operator== (const shared_ptr< _Tp > &__a, nullptr_t)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool std::operator== (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`

- `template<typename _Tp, typename _Dp >`
`bool std::operator== (nullptr_t, const unique_ptr< _Tp, _Dp > &__y)`
- `template<typename _Tp >`
`bool std::operator== (nullptr_t, const shared_ptr< _Tp > &__b)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator== (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool std::operator> (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool std::operator>= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Tp1 >`
`shared_ptr< _Tp > std::static_pointer_cast (const shared_ptr< _Tp1 > &__r)`
- `template<typename _Tp, typename _Dp >`
`void std::swap (unique_ptr< _Tp, _Dp > &__x, unique_ptr< _Tp, _Dp > &__y)`
- `template<typename _Tp >`
`void std::swap (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b)`
- `template<typename _Tp >`
`void std::swap (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b)`

2.16.1 Detailed Description

Smart pointers, etc.

2.16.2 Function Documentation

- #### 2.16.2.1 `template<typename _Tp, typename _Alloc, typename... _Args>` `shared_ptr<_Tp> std::allocate_shared (const _Alloc & __a, _Args` `&&... __args) [inline]`

Create an object that is owned by a [shared_ptr](#).

Parameters

- `__a` An allocator.
- `__args` Arguments for the `_Tp` object's constructor.

Returns

A [shared_ptr](#) that owns the newly created object.

Exceptions

An exception thrown from `_Alloc::allocate` or from the constructor of `_Tp`.

A copy of `__a` will be used to allocate memory for the `shared_ptr` and the new object.

Definition at line 528 of file `shared_ptr.h`.

```
2.16.2.2 template<typename _Del , typename _Tp , _Lock_policy _Lp>
        _Del* std::get_deleter ( const __shared_ptr< _Tp, _Lp > & __p )
        [inline]
```

2.2.3.10 `shared_ptr` `get_deleter` (experimental)

Definition at line 76 of file `shared_ptr.h`.

```
2.16.2.3 template<typename _Tp , typename... _Args> shared_ptr<_Tp>
        std::make_shared ( _Args &&... __args ) [inline]
```

Create an object that is owned by a `shared_ptr`.

Parameters

`__args` Arguments for the `_Tp` object's constructor.

Returns

A `shared_ptr` that owns the newly created object.

Exceptions

`std::bad_alloc`, or an exception thrown from the constructor of `_Tp`.

Definition at line 543 of file `shared_ptr.h`.

```
2.16.2.4 template<typename _Ch , typename _Tr , typename _Tp
        , _Lock_policy _Lp> std::basic_ostream<_Ch, _Tr>&
        std::operator<< ( std::basic_ostream<_Ch, _Tr> & __os, const
        __shared_ptr< _Tp, _Lp > & __p ) [inline]
```

2.2.3.7 `shared_ptr` I/O

Definition at line 66 of file `shared_ptr.h`.

2.17 Mutexes

Collaboration diagram for Mutexes:



Classes

- struct `std::adopt_lock_t`
Assume the calling thread has already obtained mutex ownership /// and manage it.
- struct `std::defer_lock_t`
Do not acquire ownership of the mutex.
- class `std::lock_guard< _Mutex >`
Scoped lock idiom.
- class `std::mutex`
mutex
- struct `std::once_flag`
once_flag
- class `std::recursive_mutex`
recursive_mutex
- class `std::recursive_timed_mutex`
recursive_timed_mutex
- class `std::timed_mutex`
timed_mutex
- struct `std::try_to_lock_t`
Try to acquire ownership of the mutex without blocking.

- class `std::unique_lock<_Mutex>`
unique_lock

Functions

- `mutex & std::__get_once_mutex()`
- `void std::__once_proxy()`
- `void std::__set_once_functor_lock_ptr(unique_lock<mutex> *)`
- `template<typename _Lock>`
`unique_lock<_Lock> std::__try_to_lock(_Lock &__l)`
- `template<typename _Callable, typename... _Args>`
`void std::call_once(once_flag &__once, _Callable &&__f, _Args &&...__args)`
- `template<typename _L1, typename _L2, typename... _L3>`
`void std::lock(_L1 &__l1, _L2 &__l2, _L3 &...__l3)`
- `template<typename _Mutex>`
`void std::swap(unique_lock<_Mutex> &__x, unique_lock<_Mutex> &__y)`
- `template<typename _Lock1, typename _Lock2, typename... _Lock3>`
`int std::try_lock(_Lock1 &__l1, _Lock2 &__l2, _Lock3 &...__l3)`

Variables

- `function<void()> std::__once_functor`

2.17.1 Detailed Description

Classes for mutex support.

2.17.2 Function Documentation

- 2.17.2.1** `template<typename _Callable, typename... _Args> void`
`std::call_once(once_flag & __once, _Callable && __f, _Args &&...`
`__args)`

`call_once`

Definition at line 797 of file mutex.

2.17.2.2 `template<typename _L1 , typename _L2 , typename... _L3> void
std::lock (_L1 & __l1, _L2 & __l2, _L3 &... __l3)`

Generic lock.

Parameters

`__l1` Meets Mutex requirements ([try_lock\(\)](#) may throw).

`__l2` Meets Mutex requirements ([try_lock\(\)](#) may throw).

`__l3` Meets Mutex requirements ([try_lock\(\)](#) may throw).

Exceptions

An exception thrown by an argument's [lock\(\)](#) or [try_lock\(\)](#) member.

Postcondition

All arguments are locked.

All arguments are locked via a sequence of calls to [lock\(\)](#), [try_lock\(\)](#) and [unlock\(\)](#). If the call exits via an exception any locks that were obtained will be released.

Definition at line 738 of file mutex.

References [std::lock\(\)](#).

Referenced by [std::lock\(\)](#).

2.17.2.3 `template<typename _Lock1 , typename _Lock2 , typename... _Lock3>
int std::try_lock (_Lock1 & __l1, _Lock2 & __l2, _Lock3 &... __l3
)`

Generic try_lock.

Parameters

`__l1` Meets Mutex requirements ([try_lock\(\)](#) may throw).

`__l2` Meets Mutex requirements ([try_lock\(\)](#) may throw).

`__l3` Meets Mutex requirements ([try_lock\(\)](#) may throw).

Returns

Returns -1 if all [try_lock\(\)](#) calls return true. Otherwise returns a 0-based index corresponding to the argument that returned false.

Postcondition

Either all arguments are locked, or none will be.

Sequentially calls [try_lock\(\)](#) on each argument.

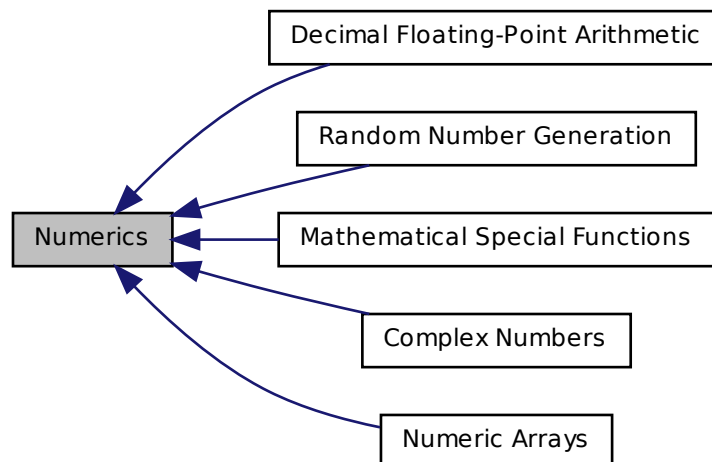
Definition at line 714 of file mutex.

References `std::try_lock()`.

Referenced by `std::try_lock()`.

2.18 Numerics

Collaboration diagram for Numerics:

**Modules**

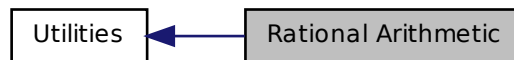
- [Complex Numbers](#)
- [Numeric Arrays](#)
- [Mathematical Special Functions](#)
- [Decimal Floating-Point Arithmetic](#)
- [Random Number Generation](#)

2.18.1 Detailed Description

Components for performing numeric operations. Includes support for complex number types, random number generation, numeric (n-at-a-time) arrays, generalized numeric algorithms, and special math functions.

2.19 Rational Arithmetic

Collaboration diagram for Rational Arithmetic:



Classes

- struct `std::ratio< _Num, _Den >`
Provides compile-time rational arithmetic.
- struct `std::ratio_add< _R1, _R2 >`
ratio_add
- struct `std::ratio_divide< _R1, _R2 >`
ratio_divide
- struct `std::ratio_equal< _R1, _R2 >`
ratio_equal
- struct `std::ratio_multiply< _R1, _R2 >`
ratio_multiply
- struct `std::ratio_not_equal< _R1, _R2 >`
ratio_not_equal
- struct `std::ratio_subtract< _R1, _R2 >`
ratio_subtract

Typedefs

- typedef ratio< num, den > **std::ratio::type**
- typedef ratio_multiply< _R1, ratio< _R2::den, _R2::num > >::type **std::ratio_divide::type**
- typedef ratio< __safe_multiply<(_R1::num/__gcd1),(_R2::num/_-gcd2)>::value, __safe_multiply<(_R1::den/__gcd2),(_R2::den/_-gcd1)>::value > **std::ratio_multiply::type**
- typedef ratio< __safe_add< __safe_multiply< _R1::num,(_R2::den/_-gcd)>::value, __safe_multiply< _R2::num,(_R1::den/_-gcd)>::value >::value, __safe_multiply< _R1::den,(_R2::den/_-gcd)>::value > **std::ratio_add::type**

Functions

- **std::ratio::static_assert** (_Num >= __INTMAX_MAX__ && _Den >= __INTMAX_MAX__, "out of range")

Variables

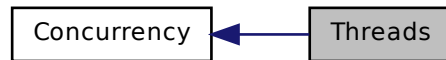
- static constexpr intmax_t **std::ratio_divide::den**
- static constexpr intmax_t **std::ratio::den**
- static constexpr intmax_t **std::ratio_add::den**
- static constexpr intmax_t **std::ratio_subtract::den**
- static constexpr intmax_t **std::ratio_multiply::den**
- static constexpr intmax_t **std::ratio_divide::num**
- static constexpr intmax_t **std::ratio::num**
- static constexpr intmax_t **std::ratio_multiply::num**
- static constexpr intmax_t **std::ratio_subtract::num**
- static constexpr intmax_t **std::ratio_add::num**
- static const intmax_t **std::__safe_add::value**
- static const intmax_t **std::__safe_multiply::value**

2.19.1 Detailed Description

Compile time representation of finite rational numbers.

2.20 Threads

Collaboration diagram for Threads:



Classes

- struct `std::hash< thread::id >`
std::hash specialization for thread::id.
- class `std::thread`
thread

Namespaces

- namespace `std::this_thread`

Functions

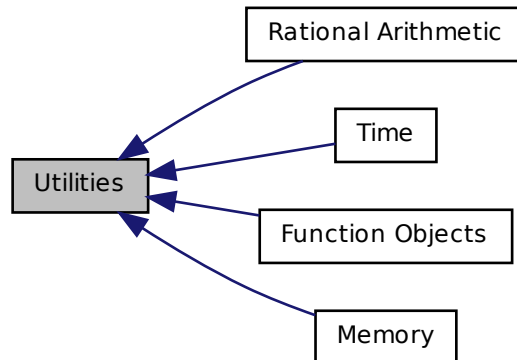
- bool **std::operator!=** (thread::id __x, thread::id __y)
- template<class _CharT, class _Traits >
 basic_ostream< _CharT, _Traits > & **std::operator<<** (basic_ostream< _CharT, _Traits > &__out, thread::id __id)
- bool **std::operator<=** (thread::id __x, thread::id __y)
- bool **std::operator>** (thread::id __x, thread::id __y)
- bool **std::operator>=** (thread::id __x, thread::id __y)
- void **std::swap** (thread &__x, thread &__y)

2.20.1 Detailed Description

Classes for thread support.

2.21 Utilities

Collaboration diagram for Utilities:



Modules

- [Time](#)
- [Memory](#)
- [Rational Arithmetic](#)
- [Function Objects](#)

2.21.1 Detailed Description

Components deemed generally useful. Includes pair, tuple, forward/move helpers, ratio, function object, metaprogramming and type traits, time, date, and memory functions.

2.22 Numeric Arrays

Collaboration diagram for Numeric Arrays:



Classes

- class `std::gslice`
Class defining multi-dimensional subset of an array.
- class `std::gslice_array< _Tp >`
Reference to multi-dimensional subset of an array.
- class `std::indirect_array< _Tp >`
Reference to arbitrary subset of an array.
- class `std::mask_array< _Tp >`
Reference to selected subset of an array.
- class `std::slice`
Class defining one-dimensional subset of an array.
- class `std::slice_array< _Tp >`
Reference to one-dimensional subset of an array.
- class `std::valarray< _Tp >`
Smart array designed to support numeric processing.

Defines

- `#define _DEFINE_BINARY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_AUGMENTED_ASSIGNMENT(_Op, _Name)`

- `#define _DEFINE_VALARRAY_EXPR_AUGMENTED_-
ASSIGNMENT(_Op, _Name)`
- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_UNARY_OPERATOR(_Op, _Name)`

Functions

- `std::gslice::gslice ()`
- `std::gslice::gslice (size_t, const valarray< size_t > &, const valarray< size_t > &)`
- `std::gslice::gslice (const gslice &)`
- `std::gslice_array::gslice_array (const gslice_array &)`
- `std::indirect_array::indirect_array (const indirect_array &)`
- `std::mask_array::mask_array (const mask_array &)`
- `std::slice::slice ()`
- `std::slice::slice (size_t, size_t, size_t)`
- `std::slice_array::slice_array (const slice_array &)`
- `std::valarray::valarray (const slice_array< _Tp > &)`
- `std::valarray::valarray (const gslice_array< _Tp > &)`
- `std::valarray::valarray (const valarray &)`
- `std::valarray::valarray (const mask_array< _Tp > &)`
- `std::valarray::valarray (const indirect_array< _Tp > &)`
- `std::valarray::valarray ()`
- `template<typename _Tp>
std::valarray::valarray (const _Tp *__restrict __p, size_t __n)`
- `std::valarray::valarray (initializer_list< _Tp >)`
- `template<class _Dom >
std::valarray::valarray (const _Expr< _Dom, _Tp > &__e)`
- `std::valarray::valarray (size_t)`
- `std::valarray::valarray (const _Tp &, size_t)`
- `std::gslice::~gslice ()`
- `_Expr< _ValFunClos< _ValArray, _Tp >, _Tp > std::valarray::apply (_Tp
func(_Tp)) const`
- `_Expr< _RefFunClos< _ValArray, _Tp >, _Tp > std::valarray::apply (_Tp
func(const _Tp &)) const`
- `template<class _Tp >
_Tp * std::begin (valarray< _Tp > &__va)`
- `template<class _Tp >
const _Tp * std::begin (const valarray< _Tp > &__va)`
- `valarray< _Tp > std::valarray::cshift (int) const`

- `template<class _Tp >`
`const _Tp * std::end (const valarray< _Tp > &__va)`
- `template<class _Tp >`
`_Tp * std::end (valarray< _Tp > &__va)`
- `_Tp std::valarray::max () const`
- `_Tp std::valarray::min () const`
- `_UnaryOp< __logical_not >::_Rt std::valarray::operator! () const`
- `template<typename _Tp >`
`_Expr< _BinClos< __not_equal_to, _ValArray, _Constant, _Tp, _Tp >, type-`
`name __fun< __not_equal_to, _Tp >::result_type > std::operator!= (const`
`valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __not_equal_to, _Constant, _ValArray, _Tp, _Tp >, type-`
`name __fun< __not_equal_to, _Tp >::result_type > std::operator!= (const _`
`Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __not_equal_to, _ValArray, _ValArray, _Tp, _Tp >, type-`
`name __fun< __not_equal_to, _Tp >::result_type > std::operator!= (const`
`valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __modulus, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __modulus, _Tp >::result_type > std::operator% (const valarray< _`
`Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __modulus, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __modulus, _Tp >::result_type > std::operator% (const valarray< _`
`Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __modulus, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __modulus, _Tp >::result_type > std::operator% (const _Tp &__t,`
`const valarray< _Tp > &__v)`
- `void std::gslice_array::operator%= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::gslice_array::operator%= (const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray::operator%= (const _Tp &)`
- `valarray< _Tp > & std::valarray::operator%= (const valarray< _Tp > &)`
- `template<class _Dom >`
`valarray< _Tp > & std::valarray::operator%= (const _Expr< _Dom, _Tp >`
`&)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_and, _ValArray, _Constant, _Tp, _Tp >, type-`
`name __fun< __bitwise_and, _Tp >::result_type > std::operator& (const`
`valarray< _Tp > &__v, const _Tp &__t)`

- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_and, _Constant, _ValArray, _Tp, _Tp >, type-`
`name __fun< __bitwise_and, _Tp >::result_type > std::operator& (const _Tp`
`&__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_and, _ValArray, _ValArray, _Tp, _Tp >, type-`
`name __fun< __bitwise_and, _Tp >::result_type > std::operator& (const`
`valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_and, _Constant, _ValArray, _Tp, _Tp >, type-`
`name __fun< __logical_and, _Tp >::result_type > std::operator&& (const _`
`Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_and, _ValArray, _Constant, _Tp, _Tp >, type-`
`name __fun< __logical_and, _Tp >::result_type > std::operator&& (const`
`valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_and, _ValArray, _ValArray, _Tp, _Tp >, type-`
`name __fun< __logical_and, _Tp >::result_type > std::operator&& (const`
`valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<class _Dom >`
`valarray< _Tp > & std::valarray::operator&= (const _Expr< _Dom, _Tp >`
`&)`
- `void std::gslice_array::operator&= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::gslice_array::operator&= (const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray::operator&= (const _Tp &)`
- `valarray< _Tp > & std::valarray::operator&= (const valarray< _Tp > &)`
- `template<typename _Tp >`
`_Expr< _BinClos< __multiplies, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __multiplies, _Tp >::result_type > std::operator* (const valarray< _`
`Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __multiplies, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __multiplies, _Tp >::result_type > std::operator* (const _Tp &__t,`
`const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __multiplies, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __multiplies, _Tp >::result_type > std::operator* (const valarray< _`
`Tp > &__v, const valarray< _Tp > &__w)`
- `void std::gslice_array::operator*= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::gslice_array::operator*= (const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray::operator*= (const valarray< _Tp > &)`
- `valarray< _Tp > & std::valarray::operator*= (const _Tp &)`

- `template<class _Dom >`
`valarray< _Tp > & std::valarray::operator*= (const _Expr< _Dom, _Tp >`
`&)`
- `template<typename _Tp >`
`_Expr< _BinClos< __plus, _ValArray, _Constant, _Tp, _Tp >, typename _`
`_fun< __plus, _Tp >::result_type > std::operator+ (const valarray< _Tp >`
`&__v, const _Tp &__t)`
- `_UnaryOp< __unary_plus >::_Rt std::valarray::operator+ () const`
- `template<typename _Tp >`
`_Expr< _BinClos< __plus, _ValArray, _ValArray, _Tp, _Tp >, typename _`
`_fun< __plus, _Tp >::result_type > std::operator+ (const valarray< _Tp >`
`&__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __plus, _Constant, _ValArray, _Tp, _Tp >, typename _`
`_fun< __plus, _Tp >::result_type > std::operator+ (const _Tp &__t, const`
`valarray< _Tp > &__v)`
- `void std::gslice_array::operator+= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::gslice_array::operator+= (const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray::operator+= (const _Tp &)`
- `valarray< _Tp > & std::valarray::operator+= (const valarray< _Tp > &)`
- `template<class _Dom >`
`valarray< _Tp > & std::valarray::operator+= (const _Expr< _Dom, _Tp >`
`&)`
- `template<typename _Tp >`
`_Expr< _BinClos< __minus, _ValArray, _ValArray, _Tp, _Tp >, typename _`
`_fun< __minus, _Tp >::result_type > std::operator- (const valarray< _Tp >`
`&__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __minus, _ValArray, _Constant, _Tp, _Tp >, typename _`
`_fun< __minus, _Tp >::result_type > std::operator- (const valarray< _Tp >`
`&__v, const _Tp &__t)`
- `_UnaryOp< __negate >::_Rt std::valarray::operator- () const`
- `template<typename _Tp >`
`_Expr< _BinClos< __minus, _Constant, _ValArray, _Tp, _Tp >, typename _`
`_fun< __minus, _Tp >::result_type > std::operator- (const _Tp &__t, const`
`valarray< _Tp > &__v)`
- `template<class _Dom >`
`void std::gslice_array::operator-= (const _Expr< _Dom, _Tp > &) const`
- `void std::gslice_array::operator-= (const valarray< _Tp > &) const`
- `valarray< _Tp > & std::valarray::operator-= (const valarray< _Tp > &)`
- `valarray< _Tp > & std::valarray::operator-= (const _Tp &)`
- `template<class _Dom >`
`valarray< _Tp > & std::valarray::operator-= (const _Expr< _Dom, _Tp >`
`&)`

- `template<typename _Tp >`
`_Expr< _BinClos< __divides, _ValArray, _ValArray, _Tp, _Tp >, typename _`
`_fun< __divides, _Tp >::result_type > std::operator/ (const valarray< _Tp >`
`&__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __divides, _ValArray, _Constant, _Tp, _Tp >, typename _`
`_fun< __divides, _Tp >::result_type > std::operator/ (const valarray< _Tp >`
`&__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __divides, _Constant, _ValArray, _Tp, _Tp >, typename _`
`_fun< __divides, _Tp >::result_type > std::operator/ (const _Tp &__t, const`
`valarray< _Tp > &__v)`
- `template<class _Dom >`
`void std::gslice_array::operator/= (const _Expr< _Dom, _Tp > &) const`
- `void std::gslice_array::operator/= (const valarray< _Tp > &) const`
- `valarray< _Tp > & std::valarray::operator/= (const _Tp &)`
- `valarray< _Tp > & std::valarray::operator/= (const valarray< _Tp > &)`
- `template<class _Dom >`
`valarray< _Tp > & std::valarray::operator/= (const _Expr< _Dom, _Tp >`
`&)`
- `template<typename _Tp >`
`_Expr< _BinClos< __less, _ValArray, _ValArray, _Tp, _Tp >, typename _`
`_fun< __less, _Tp >::result_type > std::operator< (const valarray< _Tp >`
`&__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __less, _ValArray, _Constant, _Tp, _Tp >, typename _`
`_fun< __less, _Tp >::result_type > std::operator< (const valarray< _Tp >`
`&__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __less, _Constant, _ValArray, _Tp, _Tp >, typename _`
`_fun< __less, _Tp >::result_type > std::operator< (const _Tp &__t, const`
`valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __shift_left, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __shift_left, _Tp >::result_type > std::operator<< (const valarray<`
`_Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __shift_left, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __shift_left, _Tp >::result_type > std::operator<< (const valarray<`
`_Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __shift_left, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __shift_left, _Tp >::result_type > std::operator<< (const _Tp &__t,`
`const valarray< _Tp > &__v)`

- `template<class _Dom >`
`valarray< _Tp > & std::valarray::operator<=<= (const _Expr< _Dom, _Tp`
`> &)`
- `void std::gslice_array::operator<=<= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::gslice_array::operator<=<= (const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray::operator<=<= (const _Tp &)`
- `valarray< _Tp > & std::valarray::operator<=<= (const valarray< _Tp > &)`
- `template<typename _Tp >`
`_Expr< _BinClos< __less_equal, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __less_equal, _Tp >::result_type > std::operator<=<= (const valarray<`
`_Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __less_equal, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __less_equal, _Tp >::result_type > std::operator<=<= (const valarray<`
`_Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __less_equal, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __less_equal, _Tp >::result_type > std::operator<=<= (const _Tp &__t,`
`const valarray< _Tp > &__v)`
- `void std::indirect_array::operator= (const _Tp &) const`
- `valarray< _Tp > & std::valarray::operator= (const valarray< _Tp > &)`
- `void std::slice_array::operator= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::indirect_array::operator= (const _Expr< _Dom, _Tp > &) const`
- `gslice & std::gslice::operator= (const gslice &)`
- `void std::mask_array::operator= (const _Tp &) const`
- `gslice_array & std::gslice_array::operator= (const gslice_array &)`
- `template<class _Dom >`
`void std::gslice_array::operator= (const _Expr< _Dom, _Tp > &) const`
- `void std::gslice_array::operator= (const valarray< _Tp > &) const`
- `valarray< _Tp > & std::valarray::operator= (const _Tp &)`
- `template<class _Dom >`
`valarray< _Tp > & std::valarray::operator= (const _Expr< _Dom, _Tp > &)`
- `valarray< _Tp > & std::valarray::operator= (const slice_array< _Tp > &)`
- `indirect_array & std::indirect_array::operator= (const indirect_array &)`
- `void std::indirect_array::operator= (const valarray< _Tp > &) const`
- `mask_array & std::mask_array::operator= (const mask_array &)`
- `template<class _Ex >`
`void std::mask_array::operator= (const _Expr< _Ex, _Tp > &__e) const`
- `void std::mask_array::operator= (const valarray< _Tp > &) const`
- `slice_array & std::slice_array::operator= (const slice_array &)`
- `void std::slice_array::operator= (const _Tp &) const`

- `valarray<_Tp> & std::valarray::operator= (const mask_array<_Tp> &)`
- `template<class _Dom>`
`void std::slice_array::operator= (const _Expr<_Dom, _Tp> &) const`
- `valarray<_Tp> & std::valarray::operator= (const indirect_array<_Tp> &)`
- `void std::gslice_array::operator= (const _Tp &) const`
- `valarray<_Tp> & std::valarray::operator= (const gslice_array<_Tp> &)`
- `valarray & std::valarray::operator= (initializer_list<_Tp>)`
- `template<typename _Tp>`
`_Expr<_BinClos<__equal_to, _Constant, _ValArray, _Tp, _Tp>, typename`
`__fun<__equal_to, _Tp>::result_type> std::operator== (const _Tp &__t,`
`const valarray<_Tp> &__v)`
- `template<typename _Tp>`
`_Expr<_BinClos<__equal_to, _ValArray, _Constant, _Tp, _Tp>, typename`
`__fun<__equal_to, _Tp>::result_type> std::operator== (const valarray<_`
`Tp> &__v, const _Tp &__t)`
- `template<typename _Tp>`
`_Expr<_BinClos<__equal_to, _ValArray, _ValArray, _Tp, _Tp>, typename`
`__fun<__equal_to, _Tp>::result_type> std::operator== (const valarray<_`
`Tp> &__v, const valarray<_Tp> &__w)`
- `template<typename _Tp>`
`_Expr<_BinClos<__greater, _ValArray, _Constant, _Tp, _Tp>, typename _`
`__fun<__greater, _Tp>::result_type> std::operator> (const valarray<_Tp`
`> &__v, const _Tp &__t)`
- `template<typename _Tp>`
`_Expr<_BinClos<__greater, _ValArray, _ValArray, _Tp, _Tp>, typename _`
`__fun<__greater, _Tp>::result_type> std::operator> (const valarray<_Tp`
`> &__v, const valarray<_Tp> &__w)`
- `template<typename _Tp>`
`_Expr<_BinClos<__greater, _Constant, _ValArray, _Tp, _Tp>, typename _`
`__fun<__greater, _Tp>::result_type> std::operator> (const _Tp &__t, const`
`valarray<_Tp> &__v)`
- `template<typename _Tp>`
`_Expr<_BinClos<__greater_equal, _ValArray, _Constant, _Tp, _Tp>, type-`
`name __fun<__greater_equal, _Tp>::result_type> std::operator>= (const`
`valarray<_Tp> &__v, const _Tp &__t)`
- `template<typename _Tp>`
`_Expr<_BinClos<__greater_equal, _ValArray, _ValArray, _Tp, _Tp>, type-`
`name __fun<__greater_equal, _Tp>::result_type> std::operator>= (const`
`valarray<_Tp> &__v, const valarray<_Tp> &__w)`
- `template<typename _Tp>`
`_Expr<_BinClos<__greater_equal, _Constant, _ValArray, _Tp, _Tp>, type-`
`name __fun<__greater_equal, _Tp>::result_type> std::operator>= (const`
`_Tp &__t, const valarray<_Tp> &__v)`

- `template<typename _Tp >`
`_Expr< _BinClos< __shift_right, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __shift_right, _Tp >::result_type > std::operator>> (const _Tp &__t,`
`const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __shift_right, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __shift_right, _Tp >::result_type > std::operator>> (const valarray<`
`_Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __shift_right, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __shift_right, _Tp >::result_type > std::operator>> (const valarray<`
`_Tp > &__v, const _Tp &__t)`
- `template<class _Dom >`
`void std::gslice_array::operator>>= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`
`valarray< _Tp > & std::valarray::operator>>= (const _Expr< _Dom, _Tp`
`> &)`
- `void std::gslice_array::operator>>= (const valarray< _Tp > &) const`
- `valarray< _Tp > & std::valarray::operator>>= (const valarray< _Tp > &)`
- `valarray< _Tp > & std::valarray::operator>>= (const _Tp &)`
- `slice_array< _Tp > std::valarray::operator[] (slice)`
- `_Expr< _GClos< _ValArray, _Tp >, _Tp > std::valarray::operator[] (const`
`gslice &) const`
- `valarray< _Tp > std::valarray::operator[] (const valarray< bool > &) const`
- `mask_array< _Tp > std::valarray::operator[] (const valarray< bool > &)`
- `_Expr< _IClos< _ValArray, _Tp >, _Tp > std::valarray::operator[] (const`
`valarray< size_t > &) const`
- `indirect_array< _Tp > std::valarray::operator[] (const valarray< size_t > &)`
- `_Tp & std::valarray::operator[] (size_t)`
- `gslice_array< _Tp > std::valarray::operator[] (const gslice &)`
- `const _Tp & std::valarray::operator[] (size_t) const`
- `_Expr< _SClos< _ValArray, _Tp >, _Tp > std::valarray::operator[] (slice)`
`const`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_xor, _ValArray, _ValArray, _Tp, _Tp >, type-`
`name __fun< __bitwise_xor, _Tp >::result_type > std::operator^ (const`
`valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_xor, _ValArray, _Constant, _Tp, _Tp >, type-`
`name __fun< __bitwise_xor, _Tp >::result_type > std::operator^ (const`
`valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_xor, _Constant, _ValArray, _Tp, _Tp >, type-`
`name __fun< __bitwise_xor, _Tp >::result_type > std::operator^ (const _Tp`
`&__t, const valarray< _Tp > &__v)`

- `valarray< _Tp > & std::valarray::operator^= (const _Tp &)`
- `valarray< _Tp > & std::valarray::operator^= (const valarray< _Tp > &)`
- `template<class _Dom >`
`valarray< _Tp > & std::valarray::operator^= (const _Expr< _Dom, _Tp > &)`
- `template<class _Dom >`
`void std::gslice_array::operator^= (const _Expr< _Dom, _Tp > &) const`
- `void std::gslice_array::operator^= (const valarray< _Tp > &) const`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_or, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __bitwise_or, _Tp >::result_type > std::operator| (const valarray< _`
`Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_or, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __bitwise_or, _Tp >::result_type > std::operator| (const _Tp &__t,`
`const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_or, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __bitwise_or, _Tp >::result_type > std::operator| (const valarray< _`
`Tp > &__v, const _Tp &__t)`
- `valarray< _Tp > & std::valarray::operator|= (const valarray< _Tp > &)`
- `template<class _Dom >`
`void std::gslice_array::operator|= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`
`valarray< _Tp > & std::valarray::operator|= (const _Expr< _Dom, _Tp > &)`
- `void std::gslice_array::operator|= (const valarray< _Tp > &) const`
- `valarray< _Tp > & std::valarray::operator|= (const _Tp &)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_or, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __logical_or, _Tp >::result_type > std::operator|| (const valarray< _`
`Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_or, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __logical_or, _Tp >::result_type > std::operator|| (const _Tp &__t,`
`const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_or, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __logical_or, _Tp >::result_type > std::operator|| (const valarray< _`
`Tp > &__v, const _Tp &__t)`
- `_UnaryOp< __bitwise_not >::Rt std::valarray::operator~ () const`
- `void std::valarray::resize (size_t __size, _Tp __c=_Tp())`
- `valarray< _Tp > std::valarray::shift (int) const`
- `size_t std::slice::size () const`

- `valarray< size_t > std::gslice::size () const`
- `size_t std::valarray::size () const`
- `size_t std::gslice::start () const`
- `size_t std::slice::start () const`
- `size_t std::slice::stride () const`
- `valarray< size_t > std::gslice::stride () const`
- `_Tp std::valarray::sum () const`

2.22.1 Detailed Description

Classes and functions for representing and manipulating arrays of elements.

2.22.2 Function Documentation

2.22.2.1 `std::gslice::gslice () [inline, inherited]`

Construct an empty slice.

Definition at line 149 of file `gslice.h`.

2.22.2.2 `std::gslice::gslice (size_t __o, const valarray< size_t > & __l, const valarray< size_t > & __s) [inline, inherited]`

Construct a slice.

Constructs a slice with as many dimensions as the length of the *l* and *s* arrays.

Parameters

- o* Offset in array of first element.
- l* Array of dimension lengths.
- s* Array of dimension strides between array elements.

Definition at line 153 of file `gslice.h`.

2.22.2.3 `std::gslice::gslice (const gslice & __g) [inline, inherited]`

Copy constructor.

Definition at line 158 of file `gslice.h`.

2.22.2.4 `template<typename _Tp> std::gslice_array<_Tp>::gslice_array (`
`const gslice_array<_Tp> & __a) [inline, inherited]`

Copy constructor. Both slices refer to the same underlying array.

Definition at line 144 of file gslice_array.h.

2.22.2.5 `template<typename _Tp> std::indirect_array<_Tp`
`>::indirect_array (const indirect_array<_Tp> & __a)`
`[inline, inherited]`

Copy constructor. Both slices refer to the same underlying array.

Definition at line 144 of file indirect_array.h.

2.22.2.6 `template<typename _Tp> std::mask_array<_Tp>::mask_array (`
`const mask_array<_Tp> & a) [inline, inherited]`

Copy constructor. Both slices refer to the same underlying array.

Definition at line 140 of file mask_array.h.

2.22.2.7 `std::slice::slice () [inline, inherited]`

Construct an empty slice.

Definition at line 91 of file slice_array.h.

2.22.2.8 `std::slice::slice (size_t __o, size_t __d, size_t __s) [inline,`
`inherited]`

Construct a slice.

Parameters

- o* Offset in array of first element.
- d* Number of elements in slice.
- s* Stride between array elements.

Definition at line 95 of file slice_array.h.

2.22.2.9 `template<typename _Tp> std::slice_array<_Tp>::slice_array (const slice_array<_Tp> & a) [inline, inherited]`

Copy constructor. Both slices refer to the same underlying array.

Definition at line 208 of file slice_array.h.

2.22.2.10 `template<typename _Tp> std::valarray<_Tp>::valarray (const slice_array<_Tp> & __sa) [inline, inherited]`

Construct an array with the same size and values in *sa*.

Definition at line 617 of file valarray.

2.22.2.11 `template<typename _Tp> std::valarray<_Tp>::valarray (const gslice_array<_Tp> & __ga) [inline, inherited]`

Construct an array with the same size and values in *ga*.

Definition at line 626 of file valarray.

2.22.2.12 `template<typename _Tp> std::valarray<_Tp>::valarray (const valarray<_Tp> & __v) [inline, inherited]`

Copy constructor.

Definition at line 610 of file valarray.

2.22.2.13 `template<typename _Tp> std::valarray<_Tp>::valarray (const mask_array<_Tp> & __ma) [inline, inherited]`

Construct an array with the same size and values in *ma*.

Definition at line 637 of file valarray.

2.22.2.14 `template<typename _Tp> std::valarray<_Tp>::valarray (const indirect_array<_Tp> & __ia) [inline, inherited]`

Construct an array with the same size and values in *ia*.

Definition at line 646 of file valarray.

2.22.2.15 `template<typename _Tp> std::valarray<_Tp>::valarray () [inline, inherited]`

Construct an empty array.

Definition at line 585 of file valarray.

2.22.2.16 `template<typename _Tp> std::valarray<_Tp>::valarray (initializer_list<_Tp> __l) [inline, inherited]`

Construct an array with an [initializer_list](#) of values.

Definition at line 656 of file valarray.

2.22.2.17 `template<typename _Tp> std::valarray<_Tp>::valarray (size_t __n) [inline, explicit, inherited]`

Construct an array with *n* elements.

Definition at line 589 of file valarray.

2.22.2.18 `template<typename _Tp> std::valarray<_Tp>::valarray (const _Tp & __t, size_t __n) [inline, inherited]`

Construct an array with *n* elements initialized to *t*.

Definition at line 595 of file valarray.

2.22.2.19 `std::gslice::~~gslice () [inline, inherited]`

Destructor.

Definition at line 163 of file gslice.h.

2.22.2.20 `template<class _Tp> _Expr< _ValFunClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::apply (_Tp func_Tp) const [inline, inherited]`

Apply a function to the array.

Returns a new valarray with elements assigned to the result of applying func to the corresponding element of this array. The new array has the same size as this one.

Parameters

func Function of Tp returning Tp to apply.

Returns

New valarray with transformed elements.

Definition at line 979 of file valarray.

2.22.2.21 `template<class _Tp> _Expr< _RefFunClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::apply (_Tp funcconst_Tp &) const [inline, inherited]`

Apply a function to the array.

Returns a new valarray with elements assigned to the result of applying func to the corresponding element of this array. The new array has the same size as this one.

Parameters

func Function of const Tp& returning Tp to apply.

Returns

New valarray with transformed elements.

Definition at line 987 of file valarray.

2.22.2.22 `template<class _Tp> _Tp* std::begin (valarray<_Tp> & __va)
[inline]`

Return an iterator pointing to the first element of the valarray.

Parameters

va valarray.

Definition at line 1126 of file valarray.

2.22.2.23 `template<class _Tp> const _Tp* std::begin (const valarray<_Tp>
& __va) [inline]`

Return an iterator pointing to the first element of the const valarray.

Parameters

va valarray.

Definition at line 1136 of file valarray.

2.22.2.24 `template<class _Tp> valarray<_Tp> std::valarray<_Tp>::cshift
(int __n) const [inline, inherited]`

Return a rotated array.

A new valarray is constructed as a copy of this array with elements in shifted positions. For an element with index *i*, the new position is $(i - n) \% \text{size}()$. The new valarray has the same size as the current one. Elements that are shifted beyond the array bounds are shifted into the other end of the array. No elements are lost.

Positive arguments shift toward index 0, wrapping around the top. Negative arguments shift towards the top, wrapping around to 0.

Parameters

n Number of element positions to rotate.

Returns

New valarray with elements in shifted positions.

Definition at line 905 of file valarray.

2.22.2.25 `template<class _Tp > const _Tp* std::end (const valarray<_Tp > & __va) [inline]`

Return an iterator pointing to one past the last element of the const valarray.

Parameters

va valarray.

Definition at line 1156 of file valarray.

References `std::valarray<_Tp>::size()`.

2.22.2.26 `template<class _Tp > _Tp* std::end (valarray<_Tp > & __va) [inline]`

Return an iterator pointing to one past the last element of the valarray.

Parameters

va valarray.

Definition at line 1146 of file valarray.

References `std::valarray<_Tp>::size()`.

2.22.2.27 `template<typename _Tp > _Tp std::valarray<_Tp>::max () const [inline, inherited]`

Return the maximum element using operator<().

Definition at line 971 of file valarray.

References `std::max_element()`.

2.22.2.28 `template<typename _Tp > _Tp std::valarray<_Tp>::min () const [inline, inherited]`

Return the minimum element using operator<().

Definition at line 963 of file valarray.

References `std::min_element()`.

2.22.2.29 `template<typename _Tp> valarray<_Tp>::template _UnaryOp<
__logical_not>::Rt std::valarray<_Tp>::operator! () const
[inline, inherited]`

Return a new valarray by applying unary ! to each element.

Definition at line 1006 of file valarray.

2.22.2.30 `template<typename _Tp> void std::gslice_array<_Tp
>::operator%=(const valarray<_Tp> & __v) const [inline,
inherited]`

Modulo slice elements by corresponding elements of *v*.

Definition at line 203 of file gslice_array.h.

2.22.2.31 `template<class _Tp> valarray<_Tp> & std::valarray<_Tp
>::operator%=(const _Tp & __t) [inline, inherited]`

Set each element *e* of array to *e % t*.

Definition at line 1033 of file valarray.

2.22.2.32 `template<class _Tp> valarray<_Tp> & std::valarray<_Tp
>::operator%=(const valarray<_Tp> & __v) [inline,
inherited]`

Modulo elements of array by corresponding elements of *v*.

Definition at line 1033 of file valarray.

2.22.2.33 `template<typename _Tp> void std::gslice_array<_Tp
>::operator&=(const valarray<_Tp> & __v) const [inline,
inherited]`

Logical and slice elements with corresponding elements of v .

Definition at line 207 of file `gslice_array.h`.

2.22.2.34 `template<class _Tp> valarray<_Tp> & std::valarray<_Tp>
>::operator&= (const _Tp & __t) [inline, inherited]`

Set each element e of array to $e \& t$.

Definition at line 1035 of file `valarray`.

2.22.2.35 `template<class _Tp> valarray<_Tp> & std::valarray<_Tp>
>::operator&= (const valarray<_Tp> & __v) [inline,
inherited]`

Logical and corresponding elements of v with elements of array.

Definition at line 1035 of file `valarray`.

2.22.2.36 `template<typename _Tp> void std::gslice_array<_Tp>
>::operator*= (const valarray<_Tp> & __v) const [inline,
inherited]`

Multiply slice elements by corresponding elements of v .

Definition at line 201 of file `gslice_array.h`.

2.22.2.37 `template<class _Tp> valarray<_Tp> & std::valarray<_Tp>
>::operator*= (const valarray<_Tp> & __v) [inline,
inherited]`

Multiply elements of array by corresponding elements of v .

Definition at line 1031 of file `valarray`.

2.22.2.38 `template<class _Tp> valarray<_Tp> & std::valarray<_Tp>
>::operator*= (const _Tp & __t) [inline, inherited]`

Multiply each element of array by t .

Definition at line 1031 of file valarray.

2.22.2.39 `template<typename _Tp> valarray<_Tp>::template _UnaryOp<
__unary_plus>::_Rt std::valarray<_Tp>::operator+ () const
[inline, inherited]`

Return a new valarray by applying unary + to each element.

Definition at line 1003 of file valarray.

2.22.2.40 `template<typename _Tp> void std::gslice_array<_Tp
>::operator+=(const valarray<_Tp> & __v) const [inline,
inherited]`

Add corresponding elements of v to slice elements.

Definition at line 204 of file gslice_array.h.

2.22.2.41 `template<class _Tp> valarray<_Tp> & std::valarray<_Tp
>::operator+=(const _Tp & __t) [inline, inherited]`

Add t to each element of array.

Definition at line 1029 of file valarray.

2.22.2.42 `template<class _Tp> valarray<_Tp> & std::valarray<_Tp
>::operator+=(const valarray<_Tp> & __v) [inline,
inherited]`

Add corresponding elements of v to elements of array.

Definition at line 1029 of file valarray.

2.22.2.43 `template<typename _Tp > valarray< _Tp >::template _UnaryOp<
__negate >::_Rt std::valarray< _Tp >::operator- () const
[inline, inherited]`

Return a new valarray by applying unary - to each element.

Definition at line 1004 of file valarray.

2.22.2.44 `template<typename _Tp > void std::gslice_array< _Tp
>::operator-= (const valarray< _Tp > & __v) const [inline,
inherited]`

Subtract corresponding elements of *v* from slice elements.

Definition at line 205 of file gslice_array.h.

2.22.2.45 `template<class _Tp> valarray< _Tp > & std::valarray< _Tp
>::operator-= (const valarray< _Tp > & __v) [inline,
inherited]`

Subtract corresponding elements of *v* from elements of array.

Definition at line 1030 of file valarray.

2.22.2.46 `template<class _Tp> valarray< _Tp > & std::valarray< _Tp
>::operator-= (const _Tp & __t) [inline, inherited]`

Subtract *t* to each element of array.

Definition at line 1030 of file valarray.

2.22.2.47 `template<typename _Tp > void std::gslice_array< _Tp
>::operator/= (const valarray< _Tp > & __v) const [inline,
inherited]`

Divide slice elements by corresponding elements of *v*.

Definition at line 202 of file gslice_array.h.

2.22.2.48 `template<class _Tp> valarray< _Tp > & std::valarray< _Tp
>::operator/=(const _Tp & __t) [inline, inherited]`

Divide each element of array by *t*.

Definition at line 1032 of file valarray.

2.22.2.49 `template<class _Tp> valarray< _Tp > & std::valarray< _Tp
>::operator/=(const valarray< _Tp > & __v) [inline,
inherited]`

Divide elements of array by corresponding elements of *v*.

Definition at line 1032 of file valarray.

2.22.2.50 `template<typename _Tp > void std::gslice_array< _Tp
>::operator<<=(const valarray< _Tp > & __v) const
[inline, inherited]`

Left shift slice elements by corresponding elements of *v*.

Definition at line 209 of file gslice_array.h.

2.22.2.51 `template<class _Tp> valarray< _Tp > & std::valarray< _Tp
>::operator<<=(const _Tp & __t) [inline, inherited]`

Left shift each element *e* of array by *t* bits.

Definition at line 1037 of file valarray.

2.22.2.52 `template<class _Tp> valarray< _Tp > & std::valarray< _Tp
>::operator<<=(const valarray< _Tp > & __v) [inline,
inherited]`

Left shift elements of array by corresponding elements of *v*.

Definition at line 1037 of file valarray.

2.22.2.53 `template<typename _Tp> void std::indirect_array< _Tp
>::operator=(const _Tp & __t) const [inline, inherited]`

Assign all slice elements to *t*.

Definition at line 164 of file indirect_array.h.

2.22.2.54 `template<typename _Tp> valarray< _Tp > & std::valarray< _Tp
>::operator=(const valarray< _Tp > & __v) [inline,
inherited]`

Assign elements to an array.

Assign elements of array to values in *v*. Results are undefined if *v* does not have the same size as this array.

Parameters

v Valarray to get values from.

Definition at line 677 of file valarray.

2.22.2.55 `template<typename _Tp> void std::slice_array< _Tp >::operator=
(const valarray< _Tp > & __v) const [inline, inherited]`

Assign slice elements to corresponding elements of *v*.

Definition at line 230 of file slice_array.h.

2.22.2.56 `gslice & std::gslice::operator=(const gslice & __g) [inline,
inherited]`

Assignment operator.

Definition at line 170 of file gslice.h.

2.22.2.57 `template<typename _Tp> void std::mask_array< _Tp >::operator=
(const _Tp & __t) const [inline, inherited]`

Assign all slice elements to *t*.

Definition at line 159 of file mask_array.h.

2.22.2.58 `template<typename _Tp > gslice_array< _Tp > &
std::gslice_array< _Tp >::operator= (const gslice_array< _Tp > &
__a) [inline, inherited]`

Assignment operator. Assigns slice elements to corresponding /// elements of *a*.

Definition at line 149 of file gslice_array.h.

References `std::valarray< _Tp >::size()`.

2.22.2.59 `template<typename _Tp > void std::gslice_array< _Tp >::operator=
(const valarray< _Tp > & __v) const [inline, inherited]`

Assign slice elements to corresponding elements of *v*.

Definition at line 167 of file gslice_array.h.

References `std::valarray< _Tp >::size()`.

2.22.2.60 `template<typename _Tp> valarray< _Tp > & std::valarray< _Tp
>::operator= (const _Tp & __t) [inline, inherited]`

Assign elements to a value.

Assign all elements of array to *t*.

Parameters

t Value for elements.

Definition at line 725 of file valarray.

2.22.2.61 `template<typename _Tp> valarray< _Tp > & std::valarray< _Tp
>::operator= (const slice_array< _Tp > & __sa) [inline,
inherited]`

Assign elements to an array subset.

Assign elements of array to values in *sa*. Results are undefined if *sa* does not have the same size as this array.

Parameters

sa Array slice to get values from.

Definition at line 733 of file valarray.

2.22.2.62 `template<typename _Tp > indirect_array< _Tp > &
std::indirect_array< _Tp >::operator= (const indirect_array< _Tp
> & __a) [inline, inherited]`

Assignment operator. Assigns elements to corresponding elements /// of *a*.

Definition at line 155 of file indirect_array.h.

2.22.2.63 `template<typename _Tp > void std::indirect_array< _Tp
>::operator= (const valarray< _Tp > & __v) const [inline,
inherited]`

Assign slice elements to corresponding elements of *v*.

Definition at line 169 of file indirect_array.h.

2.22.2.64 `template<typename _Tp > mask_array< _Tp > &
std::mask_array< _Tp >::operator= (const mask_array< _Tp > &
__a) [inline, inherited]`

Assignment operator. Assigns elements to corresponding elements /// of *a*.

Definition at line 150 of file mask_array.h.

2.22.2.65 `template<typename _Tp > slice_array< _Tp > & std::slice_array<
_Tp >::operator= (const slice_array< _Tp > & __a) [inline,
inherited]`

Assignment operator. Assigns slice elements to corresponding *///* elements of *a*.

Definition at line 216 of file slice_array.h.

2.22.2.66 `template<typename _Tp> void std::slice_array<_Tp>::operator=
(const _Tp & __t) const [inline, inherited]`

Assign all slice elements to *t*.

Definition at line 225 of file slice_array.h.

2.22.2.67 `template<typename _Tp> valarray<_Tp> & std::valarray<_Tp>
>::operator= (const mask_array<_Tp> & __ma) [inline,
inherited]`

Assign elements to an array subset.

Assign elements of array to values in *ma*. Results are undefined if *ma* does not have the same size as this array.

Parameters

ma Array slice to get values from.

Definition at line 753 of file valarray.

2.22.2.68 `template<typename _Tp> valarray<_Tp> & std::valarray<_Tp>
>::operator= (const indirect_array<_Tp> & __ia) [inline,
inherited]`

Assign elements to an array subset.

Assign elements of array to values in *ia*. Results are undefined if *ia* does not have the same size as this array.

Parameters

ia Array slice to get values from.

Definition at line 763 of file valarray.

2.22.2.69 `template<typename _Tp> void std::gslice_array<_Tp>::operator=(const _Tp & __t) const [inline, inherited]`

Assign all slice elements to *t*.

Definition at line 159 of file `gslice_array.h`.

References `std::valarray<_Tp>::size()`.

2.22.2.70 `template<typename _Tp> valarray<_Tp> & std::valarray<_Tp>::operator=(const gslice_array<_Tp> & __ga) [inline, inherited]`

Assign elements to an array subset.

Assign elements of array to values in *ga*. Results are undefined if *ga* does not have the same size as this array.

Parameters

ga Array slice to get values from.

Definition at line 743 of file `valarray`.

References `std::valarray<_Tp>::size()`.

2.22.2.71 `template<typename _Tp> valarray<_Tp> & std::valarray<_Tp>::operator=(initializer_list<_Tp> __l) [inline, inherited]`

Assign elements to an [initializer_list](#).

Assign elements of array to values in *l*. Results are undefined if *l* does not have the same size as this array.

Parameters

l [initializer_list](#) to get values from.

Definition at line 701 of file `valarray`.

2.22.2.72 `template<typename _Tp> void std::gslice_array< _Tp
>::operator>>= (const valarray< _Tp> & __v) const
[inline, inherited]`

Right shift slice elements by corresponding elements of *v*.

Definition at line 210 of file `gslice_array.h`.

2.22.2.73 `template<class _Tp> valarray< _Tp> & std::valarray< _Tp
>::operator>>= (const valarray< _Tp> & __v) [inline,
inherited]`

Right shift elements of array by corresponding elements of *v*.

Definition at line 1038 of file `valarray`.

2.22.2.74 `template<class _Tp> valarray< _Tp> & std::valarray< _Tp
>::operator>>= (const _Tp & __t) [inline, inherited]`

Right shift each element *e* of array by *t* bits.

Definition at line 1038 of file `valarray`.

2.22.2.75 `template<typename _Tp> slice_array< _Tp> std::valarray< _Tp
>::operator[] (slice __s) [inline, inherited]`

Return a reference to an array subset.

Returns a new `valarray` containing the elements of the array indicated by the slice argument. The new `valarray` has the same size as the input slice.

See also

[slice](#).

Parameters

s The source slice.

Returns

New `valarray` containing elements in *s*.

Definition at line 790 of file valarray.

2.22.2.76 `template<typename _Tp > _Expr< _GClos< _ValArray, _Tp >, _Tp
> std::valarray< _Tp >::operator[] (const gslice & __gs) const
[inline, inherited]`

Return an array subset.

Returns a [slice_array](#) referencing the elements of the array indicated by the slice argument.

See also

[gslice](#).

Parameters

s The source slice.

Returns

[Slice_array](#) referencing elements indicated by *s*.

Definition at line 795 of file valarray.

2.22.2.77 `template<typename _Tp > valarray< _Tp > std::valarray< _Tp
>::operator[] (const valarray< bool > & __m) const [inline,
inherited]`

Return an array subset.

Returns a new valarray containing the elements of the array indicated by the argument. The input is a valarray of bool which represents a bitmask indicating which elements should be copied into the new valarray. Each element of the array is added to the return valarray if the corresponding element of the argument is true.

Parameters

m The valarray bitmask.

Returns

New valarray containing elements indicated by *m*.

Definition at line 812 of file valarray.

References `std::valarray< _Tp >::size()`.

2.22.2.78 `template<typename _Tp > mask_array< _Tp > std::valarray< _Tp >::operator[] (const valarray< bool > & __m) [inline, inherited]`

Return a reference to an array subset.

Returns a new `mask_array` referencing the elements of the array indicated by the argument. The input is a valarray of bool which represents a bitmask indicating which elements are part of the subset. Elements of the array are part of the subset if the corresponding element of the argument is true.

Parameters

m The valarray bitmask.

Returns

New valarray containing elements indicated by *m*.

Definition at line 824 of file valarray.

References `std::valarray< _Tp >::size()`.

2.22.2.79 `template<typename _Tp > _Expr< _IClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::operator[] (const valarray< size_t > & __i) const [inline, inherited]`

Return an array subset.

Returns a new valarray containing the elements of the array indicated by the argument. The elements in the argument are interpreted as the indices of elements of this valarray to copy to the return valarray.

Parameters

i The valarray element index list.

Returns

New valarray containing elements in *s*.

Definition at line 835 of file valarray.

2.22.2.80 `template<typename _Tp> indirect_array<_Tp> std::valarray<_Tp>::operator[] (const valarray<size_t> & __i) [inline, inherited]`

Return a reference to an array subset.

Returns an [indirect_array](#) referencing the elements of the array indicated by the argument. The elements in the argument are interpreted as the indices of elements of this valarray to include in the subset. The returned [indirect_array](#) refers to these elements.

Parameters

i The valarray element index list.

Returns

Indirect_array referencing elements in *i*.

Definition at line 843 of file valarray.

References std::valarray<_Tp>::size().

2.22.2.81 `template<typename _Tp> _Tp & std::valarray<_Tp>::operator[] (size_t __i) [inline, inherited]`

Return a reference to the *i*'th array element.

Parameters

i Index of element to return.

Returns

Reference to the *i*'th element.

Definition at line 556 of file valarray.

2.22.2.82 `template<typename _Tp> gslice_array<_Tp> std::valarray<_Tp>::operator[] (const gslice & __gs) [inline, inherited]`

Return a reference to an array subset.

Returns a new valarray containing the elements of the array indicated by the gslice argument. The new valarray has the same size as the input gslice.

See also

[gslice](#).

Parameters

s The source gslice.

Returns

New valarray containing elements in *s*.

Definition at line 804 of file valarray.

2.22.2.83 `template<typename _Tp> _Expr<_SClos<_ValArray, _Tp>, _Tp> std::valarray<_Tp>::operator[](slice __s) const [inline, inherited]`

Return an array subset.

Returns a new valarray containing the elements of the array indicated by the slice argument. The new valarray has the same size as the input slice.

See also

[slice](#).

Parameters

s The source slice.

Returns

New valarray containing elements in *s*.

Definition at line 782 of file valarray.

2.22.2.84 `template<class _Tp> valarray<_Tp> & std::valarray<_Tp>::operator^=(const _Tp & __t) [inline, inherited]`

Set each element *e* of array to $e \wedge t$.

Definition at line 1034 of file valarray.

2.22.2.85 `template<class _Tp> valarray<_Tp> & std::valarray<_Tp>::operator^= (const valarray<_Tp> & __v) [inline, inherited]`

Logical xor corresponding elements of v with elements of array.

Definition at line 1034 of file valarray.

2.22.2.86 `template<typename _Tp> void std::gslice_array<_Tp>::operator^= (const valarray<_Tp> & __v) const [inline, inherited]`

Logical xor slice elements with corresponding elements of v .

Definition at line 206 of file gslice_array.h.

2.22.2.87 `template<class _Tp> valarray<_Tp> & std::valarray<_Tp>::operator|= (const valarray<_Tp> & __v) [inline, inherited]`

Logical or corresponding elements of v with elements of array.

Definition at line 1036 of file valarray.

2.22.2.88 `template<typename _Tp> void std::gslice_array<_Tp>::operator|= (const valarray<_Tp> & __v) const [inline, inherited]`

Logical or slice elements with corresponding elements of v .

Definition at line 208 of file gslice_array.h.

2.22.2.89 `template<class _Tp> valarray<_Tp> & std::valarray<_Tp>::operator|= (const _Tp & __t) [inline, inherited]`

Set each element e of array to $e \mid t$.

Definition at line 1036 of file valarray.

2.22.2.90 `template<typename _Tp> valarray<_Tp>::template _UnaryOp<
__bitwise_not>::_Rt std::valarray<_Tp>::operator~() const
[inline, inherited]`

Return a new valarray by applying unary ~ to each element.

Definition at line 1005 of file valarray.

2.22.2.91 `template<class _Tp> void std::valarray<_Tp>::resize (size_t
__size, _Tp __c = _Tp()) [inline, inherited]`

Resize array.

Resize this array to *size* and set all elements to *c*. All references and iterators are invalidated.

Parameters

size New array size.

c New value for all elements.

Definition at line 946 of file valarray.

2.22.2.92 `template<class _Tp> valarray<_Tp> std::valarray<_Tp>::shift
(int __n) const [inline, inherited]`

Return a shifted array.

A new valarray is constructed as a copy of this array with elements in shifted positions. For an element with index *i*, the new position is *i* - *n*. The new valarray has the same size as the current one. New elements without a value are set to 0. Elements whose new position is outside the bounds of the array are discarded.

Positive arguments shift toward index 0, discarding elements [0, *n*). Negative arguments discard elements from the top of the array.

Parameters

n Number of element positions to shift.

Returns

New valarray with elements in shifted positions.

Definition at line 864 of file valarray.

2.22.2.93 `size_t std::slice::size () const [inline, inherited]`

Return size of slice.

Definition at line 103 of file slice_array.h.

2.22.2.94 `valarray< size_t > std::gslice::size () const [inline, inherited]`

Return array of sizes of slice dimensions.

Definition at line 139 of file gslice.h.

2.22.2.95 `template<class _Tp > size_t std::valarray< _Tp >::size () const [inline, inherited]`

Return the number of elements in array.

Definition at line 851 of file valarray.

Referenced by `std::end()`, `std::valarray< _Tp >::operator=()`, `std::gslice_array< _Tp >::operator=()`, and `std::valarray< _Tp >::operator[]()`.

2.22.2.96 `size_t std::gslice::start () const [inline, inherited]`

Return array offset of first slice element.

Definition at line 135 of file gslice.h.

2.22.2.97 `size_t std::slice::start () const [inline, inherited]`

Return array offset of first slice element.

Definition at line 99 of file slice_array.h.

2.22.2.98 `size_t std::slice::stride () const [inline, inherited]`

Return array stride of slice.

Definition at line 107 of file slice_array.h.

2.22.2.99 `valarray< size_t > std::gslice::stride () const [inline, inherited]`

Return array of array strides for each dimension.

Definition at line 143 of file gslice.h.

2.22.2.100 `template<class _Tp > _Tp std::valarray< _Tp >::sum () const [inline, inherited]`

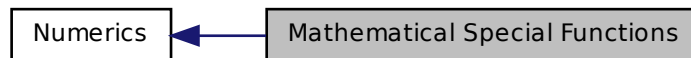
Return the sum of all elements in the array.

Accumulates the sum of all elements into a `Tp` using `+=`. The order of adding the elements is unspecified.

Definition at line 856 of file valarray.

2.23 Mathematical Special Functions

Collaboration diagram for Mathematical Special Functions:



Functions

- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_laguerre` (unsigned int __n, unsigned int __m, _Tp __x)
- `float std::tr1::assoc_laguerref` (unsigned int __n, unsigned int __m, float __x)
- `long double std::tr1::assoc_laguerrel` (unsigned int __n, unsigned int __m, long double __x)
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_legendre` (unsigned int __l, unsigned int __m, _Tp __x)
- `float std::tr1::assoc_legendref` (unsigned int __l, unsigned int __m, float __x)
- `long double std::tr1::assoc_legendrel` (unsigned int __l, unsigned int __m, long double __x)
- `template<typename _Tpx, typename _Tpy >`
`__gnu_cxx::__promote_2< _Tpx, _Tpy >::__type std::tr1::beta` (_Tpx __x, _Tpy __y)
- `float std::tr1::betaf` (float __x, float __y)
- `long double std::tr1::betal` (long double __x, long double __y)
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::tr1::comp_ellint_1` (_Tp __k)
- `float std::tr1::comp_ellint_1f` (float __k)
- `long double std::tr1::comp_ellint_1l` (long double __k)
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::tr1::comp_ellint_2` (_Tp __k)
- `float std::tr1::comp_ellint_2f` (float __k)
- `long double std::tr1::comp_ellint_2l` (long double __k)
- `template<typename _Tp, typename _Tpn >`
`__gnu_cxx::__promote_2< _Tp, _Tpn >::__type std::tr1::comp_ellint_3` (_Tp __k, _Tpn __nu)
- `float std::tr1::comp_ellint_3f` (float __k, float __nu)
- `long double std::tr1::comp_ellint_3l` (long double __k, long double __nu)
- `template<typename _Tpa, typename _Tpc, typename _Tp >`
`__gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type std::tr1::conf_hyperg` (_Tpa __a, _Tpc __c, _Tp __x)
- `float std::tr1::conf_hypergf` (float __a, float __c, float __x)
- `long double std::tr1::conf_hypergl` (long double __a, long double __c, long double __x)
- `template<typename _Tpnu, typename _Tp >`
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_i` (_Tpnu __nu, _Tp __x)
- `float std::tr1::cyl_bessel_if` (float __nu, float __x)
- `long double std::tr1::cyl_bessel_il` (long double __nu, long double __x)

- `template<typename _Tpnu, typename _Tp >`
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_j (_Tpnu`
`__nu, _Tp __x)`
- `float std::tr1::cyl_bessel_jf (float __nu, float __x)`
- `long double std::tr1::cyl_bessel_jl (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp >`
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_k (_`
`Tpnu __nu, _Tp __x)`
- `float std::tr1::cyl_bessel_kf (float __nu, float __x)`
- `long double std::tr1::cyl_bessel_kl (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp >`
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_neumann (_`
`Tpnu __nu, _Tp __x)`
- `float std::tr1::cyl_neumannf (float __nu, float __x)`
- `long double std::tr1::cyl_neumannl (long double __nu, long double __x)`
- `template<typename _Tp, typename _Tpp >`
`__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::tr1::ellint_1 (_Tp __k, _`
`Tpp __phi)`
- `float std::tr1::ellint_1f (float __k, float __phi)`
- `long double std::tr1::ellint_1l (long double __k, long double __phi)`
- `template<typename _Tp, typename _Tpp >`
`__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::tr1::ellint_2 (_Tp __k, _`
`Tpp __phi)`
- `float std::tr1::ellint_2f (float __k, float __phi)`
- `long double std::tr1::ellint_2l (long double __k, long double __phi)`
- `template<typename _Tp, typename _Tpn, typename _Tpp >`
`__gnu_cxx::__promote_3< _Tp, _Tpn, _Tpp >::__type std::tr1::ellint_3 (_Tp`
`__k, _Tpn __nu, _Tpp __phi)`
- `float std::tr1::ellint_3f (float __k, float __nu, float __phi)`
- `long double std::tr1::ellint_3l (long double __k, long double __nu, long double`
`__phi)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::tr1::expint (_Tp __x)`
- `float std::tr1::expintf (float __x)`
- `long double std::tr1::expintl (long double __x)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::tr1::hermite (unsigned int __n, _Tp`
`__x)`
- `float std::tr1::hermitef (unsigned int __n, float __x)`
- `long double std::tr1::hermitel (unsigned int __n, long double __x)`
- `template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp >`
`__gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type std::tr1::hyperg`
`(_Tpa __a, _Tpb __b, _Tpc __c, _Tp __x)`
- `float std::tr1::hypergf (float __a, float __b, float __c, float __x)`

- long double **std::tr1::hypergl** (long double __a, long double __b, long double __c, long double __x)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **std::tr1::laguerre** (unsigned int __n, _Tp __x)
- float **std::tr1::laguerref** (unsigned int __n, float __x)
- long double **std::tr1::laguerrel** (unsigned int __n, long double __x)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **std::tr1::legendre** (unsigned int __n, _Tp __x)
- float **std::tr1::legendref** (unsigned int __n, float __x)
- long double **std::tr1::legendrel** (unsigned int __n, long double __x)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **std::tr1::riemann_zeta** (_Tp __x)
- float **std::tr1::riemann_zetaf** (float __x)
- long double **std::tr1::riemann_zetal** (long double __x)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **std::tr1::sph_bessel** (unsigned int __n, _Tp __x)
- float **std::tr1::sph_besself** (unsigned int __n, float __x)
- long double **std::tr1::sph_bessell** (unsigned int __n, long double __x)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **std::tr1::sph_legendre** (unsigned int __l, unsigned int __m, _Tp __theta)
- float **std::tr1::sph_legendref** (unsigned int __l, unsigned int __m, float __theta)
- long double **std::tr1::sph_legendrel** (unsigned int __l, unsigned int __m, long double __theta)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **std::tr1::sph_neumann** (unsigned int __n, _Tp __x)
- float **std::tr1::sph_neumannf** (unsigned int __n, float __x)
- long double **std::tr1::sph_neumannl** (unsigned int __n, long double __x)

2.23.1 Detailed Description

A collection of advanced mathematical special functions.

2.23.2 Function Documentation

2.23.2.1 `template<typename _Tp> __gnu_cxx::__promote<_Tp>::__type
std::tr1::assoc_laguerre (unsigned int __n, unsigned int __m, _Tp
__x) [inline]`

5.2.1.1 Associated Laguerre polynomials.

Definition at line 1081 of file tr1/cmath.

2.23.2.2 `template<typename _Tp> __gnu_cxx::__promote<_Tp>::__type
std::tr1::assoc_legendre (unsigned int __l, unsigned int __m, _Tp
__x) [inline]`

5.2.1.2 Associated Legendre functions.

Definition at line 1098 of file tr1/cmath.

2.23.2.3 `template<typename _Tpx, typename _Tpy>
__gnu_cxx::__promote_2<_Tpx, _Tpy>::__type std::tr1::beta (_Tpx
__x, _Tpy __y) [inline]`

5.2.1.3 Beta functions.

Definition at line 1115 of file tr1/cmath.

2.23.2.4 `template<typename _Tp> __gnu_cxx::__promote<_Tp>::__type
std::tr1::comp_ellint_1 (_Tp __k) [inline]`

5.2.1.4 Complete elliptic integrals of the first kind.

Definition at line 1132 of file tr1/cmath.

2.23.2.5 `template<typename _Tp> __gnu_cxx::__promote<_Tp>::__type
std::tr1::comp_ellint_2 (_Tp __k) [inline]`

5.2.1.5 Complete elliptic integrals of the second kind.

Definition at line 1149 of file tr1/cmath.

```
2.23.2.6  template<typename _Tp , typename _Tpn > __gnu_cxx::__promote_2<_Tp, _Tpn>::__type std::tr1::comp_ellint_3( _Tp __k, _Tpn __nu ) [inline]
```

5.2.1.6 Complete elliptic integrals of the third kind.

Definition at line 1166 of file tr1/cmath.

```
2.23.2.7  template<typename _Tpa , typename _Tpc , typename _Tp >
          > __gnu_cxx::__promote_3<_Tpa, _Tpc, _Tp>::__type
          std::tr1::conf_hyperg( _Tpa __a, _Tpc __c, _Tp __x ) [inline]
```

5.2.1.7 Confluent hypergeometric functions.

Definition at line 1183 of file tr1/cmath.

```
2.23.2.8  template<typename _Tpnu , typename _Tp >
          __gnu_cxx::__promote_2<_Tpnu, _Tp>::__type std::tr1::cyl_bessel_i
          ( _Tpnu __nu, _Tp __x ) [inline]
```

5.2.1.8 Regular modified cylindrical Bessel functions.

Definition at line 1200 of file tr1/cmath.

```
2.23.2.9  template<typename _Tpnu , typename _Tp >
          __gnu_cxx::__promote_2<_Tpnu, _Tp>::__type std::tr1::cyl_bessel_j
          ( _Tpnu __nu, _Tp __x ) [inline]
```

5.2.1.9 Cylindrical Bessel functions (of the first kind).

Definition at line 1217 of file tr1/cmath.

2.23.2.10 `template<typename _Tpnu , typename _Tp >`
`__gnu_cxx::__promote_2<_Tpnu, _Tp>::__type`
`std::tr1::cyl_bessel_k (_Tpnu __nu, _Tp __x) [inline]`

5.2.1.10 Irregular modified cylindrical Bessel functions.

Definition at line 1234 of file tr1/cmath.

2.23.2.11 `template<typename _Tpnu , typename _Tp >`
`__gnu_cxx::__promote_2<_Tpnu, _Tp>::__type`
`std::tr1::cyl_neumann (_Tpnu __nu, _Tp __x) [inline]`

5.2.1.11 Cylindrical Neumann functions.

Definition at line 1251 of file tr1/cmath.

2.23.2.12 `template<typename _Tp , typename _Tpp >`
`__gnu_cxx::__promote_2<_Tp, _Tpp>::__type std::tr1::ellint_1 (`
`_Tp __k, _Tpp __phi) [inline]`

5.2.1.12 Incomplete elliptic integrals of the first kind.

Definition at line 1268 of file tr1/cmath.

2.23.2.13 `template<typename _Tp , typename _Tpp >`
`__gnu_cxx::__promote_2<_Tp, _Tpp>::__type std::tr1::ellint_2 (`
`_Tp __k, _Tpp __phi) [inline]`

5.2.1.13 Incomplete elliptic integrals of the second kind.

Definition at line 1285 of file tr1/cmath.

2.23.2.14 `template<typename _Tp , typename _Tpn , typename _Tpp`
`> __gnu_cxx::__promote_3<_Tp, _Tpn, _Tpp>::__type`
`std::tr1::ellint_3 (_Tp __k, _Tpn __nu, _Tpp __phi) [inline]`

5.2.1.14 Incomplete elliptic integrals of the third kind.

Definition at line 1302 of file tr1/cmath.

2.23.2.15 `template<typename _Tp> __gnu_cxx::__promote<_Tp>::__type
std::tr1::expint (_Tp __x) [inline]`

5.2.1.15 Exponential integrals.

Definition at line 1319 of file tr1/cmath.

2.23.2.16 `template<typename _Tp> __gnu_cxx::__promote<_Tp>::__type
std::tr1::hermite (unsigned int __n, _Tp __x) [inline]`

5.2.1.16 Hermite polynomials.

Definition at line 1336 of file tr1/cmath.

2.23.2.17 `template<typename _Tpa, typename _Tpb, typename _Tpc,
typename _Tp> __gnu_cxx::__promote_4<_Tpa, _Tpb, _Tpc,
_Tp>::__type std::tr1::hyperg (_Tpa __a, _Tpb __b, _Tpc __c,
_Tp __x) [inline]`

5.2.1.17 Hypergeometric functions.

Definition at line 1353 of file tr1/cmath.

2.23.2.18 `template<typename _Tp> __gnu_cxx::__promote<_Tp>::__type
std::tr1::laguerre (unsigned int __n, _Tp __x) [inline]`

5.2.1.18 Laguerre polynomials.

Definition at line 1370 of file tr1/cmath.

2.23.2.19 `template<typename _Tp> __gnu_cxx::__promote<_Tp>::__type
std::tr1::legendre (unsigned int __n, _Tp __x) [inline]`

5.2.1.19 Legendre polynomials.

Definition at line 1387 of file tr1/cmath.

2.23.2.20 `template<typename _Tp> __gnu_cxx::__promote<_Tp>::__type
std::tr1::riemann_zeta (_Tp __x) [inline]`

5.2.1.20 Riemann zeta function.

Definition at line 1404 of file tr1/cmath.

2.23.2.21 `template<typename _Tp> __gnu_cxx::__promote<_Tp>::__type
std::tr1::sph_bessel (unsigned int __n, _Tp __x) [inline]`

5.2.1.21 Spherical Bessel functions.

Definition at line 1421 of file tr1/cmath.

2.23.2.22 `template<typename _Tp> __gnu_cxx::__promote<_Tp>::__type
std::tr1::sph_legendre (unsigned int __l, unsigned int __m, _Tp
__theta) [inline]`

5.2.1.22 Spherical associated Legendre functions.

Definition at line 1438 of file tr1/cmath.

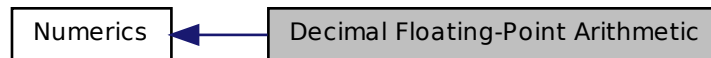
2.23.2.23 `template<typename _Tp> __gnu_cxx::__promote<_Tp>::__type
std::tr1::sph_neumann (unsigned int __n, _Tp __x) [inline]`

5.2.1.23 Spherical Neumann functions.

Definition at line 1455 of file tr1/cmath.

2.24 Decimal Floating-Point Arithmetic

Collaboration diagram for Decimal Floating-Point Arithmetic:



Namespaces

- namespace `std::decimal`

2.24.1 Detailed Description

Classes and functions for decimal floating-point arithmetic.

2.25 Binder Classes

Collaboration diagram for Binder Classes:



Classes

- class `std::binder1st<_Operation>`
One of the `binder` functors.
- class `std::binder2nd<_Operation>`

One of the *binder functors*.

- struct `std::is_bind_expression< _Tp >`
Determines if the given type `_Tp` is a function object should be treated as a subexpression when evaluating calls to function objects returned by `bind()`. [TR1 3.6.1].
- struct `std::is_bind_expression< _Bind< _Signature > >`
Class template `_Bind` is always a bind expression.
- struct `std::is_bind_expression< _Bind_result< _Result, _Signature > >`
Class template `_Bind` is always a bind expression.
- struct `std::is_placeholder< _Tp >`
Determines if the given type `_Tp` is a placeholder in a `bind()` expression and, if so, which placeholder it is. [TR1 3.6.2].
- struct `std::is_placeholder< _Placeholder< _Num > >`

Namespaces

- namespace `std::placeholders`

Functions

- template<typename _Functor, typename... _ArgTypes>
`_Bind_helper< _Functor, _ArgTypes...>::type` `std::bind` (`_Functor &&__f, _ArgTypes &&...__args`)
- template<typename _Operation, typename _Tp >
`binder1st< _Operation >` `std::bind1st` (`const _Operation &__fn, const _Tp &__x`)
- template<typename _Operation, typename _Tp >
`binder2nd< _Operation >` `std::bind2nd` (`const _Operation &__fn, const _Tp &__x`)

2.25.1 Detailed Description

Binders turn functions/functors with two arguments into functors with a single argument, storing an argument to be applied later. For example, a variable `B` of type `binder1st` is constructed from a functor `f` and an argument `x`. Later, `B's operator()` is called with a single argument `y`. The return value is the value of `f(x, y)`. `B` can be *called* with various arguments (`y1, y2, ...`) and will in turn call `f(x, y1), f(x, y2), ...`

The function `bind1st` is provided to save some typing. It takes the function and an argument as parameters, and returns an instance of `binder1st`.

The type `binder2nd` and its creator function `bind2nd` do the same thing, but the stored argument is passed as the second parameter instead of the first, e.g., `bind2nd(std::minus<float>, 1.3)` will create a functor whose `operator()` accepts a floating-point number, subtracts 1.3 from it, and returns the result. (If `bind1st` had been used, the functor would perform $1.3 - x$ instead.

Creator-wrapper functions like `bind1st` are intended to be used in calling algorithms. Their return values will be temporary objects. (The goal is to not require you to type names like `std::binder1st<std::plus<int>>` for declaring a variable to hold the return value from `bind1st(std::plus<int>, 5)`).

These become more useful when combined with the composition functions.

2.25.2 Function Documentation

2.25.2.1 `template<typename _Function , typename... _ArgTypes>
_Bind_helper<_Function, _ArgTypes...>::type std::bind (_Function
&& __f, _ArgTypes &&... __args) [inline]`

Function template for `std::bind`.

Definition at line 1435 of file `functional`.

Referenced by `std::is_permutation()`.

2.25.2.2 `template<typename _Operation , typename _Tp >
binder1st<_Operation> std::bind1st (const _Operation & __fn,
const _Tp & __x) [inline]`

One of the [binder functors](#).

Definition at line 127 of file `binders.h`.

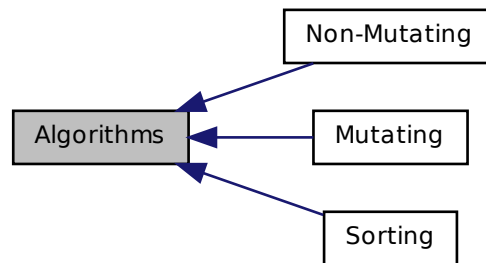
2.25.2.3 `template<typename _Operation , typename _Tp >
binder2nd<_Operation> std::bind2nd (const _Operation & __fn,
const _Tp & __x) [inline]`

One of the [binder functors](#).

Definition at line 162 of file `binders.h`.

2.26 Algorithms

Collaboration diagram for Algorithms:



Modules

- [Mutating](#)
- [Non-Mutating](#)
- [Sorting](#)

2.26.1 Detailed Description

Components for performing algorithmic operations. Includes non-modifying sequence, modifying (mutating) sequence, sorting, searching, merge, partition, heap, set, minima, maxima, and permutation operations.

2.27 Mutating

Collaboration diagram for Mutating:



Functions

- `template<typename _II, typename _OI >`
`_OI std::copy (_II __first, _II __last, _OI __result)`
- `template<typename _BI1, typename _BI2 >`
`_BI2 std::copy_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::copy_if (_InputIterator __first, _InputIterator __last, _-`
`OutputIterator __result, _Predicate __pred)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`
`_OutputIterator std::copy_n (_InputIterator __first, _Size __n, _OutputIterator`
`__result)`
- `template<typename _ForwardIterator, typename _Tp >`
`void std::fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__-`
`value)`
- `template<typename _OI, typename _Size, typename _Tp >`
`_OI std::fill_n (_OI __first, _Size __n, const _Tp &__value)`
- `template<typename _ForwardIterator, typename _Generator >`
`void std::generate (_ForwardIterator __first, _ForwardIterator __last, _Generator`
`__gen)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`
`_OutputIterator std::generate_n (_OutputIterator __first, _Size __n, _Generator`
`__gen)`
- `template<typename _InputIterator, typename _Predicate >`
`bool std::is_partitioned (_InputIterator __first, _InputIterator __last, _Predicate`
`__pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`void std::iter_swap (_ForwardIterator1 __a, _ForwardIterator2 __b)`
- `template<typename _II, typename _OI >`
`_OI std::move (_II __first, _II __last, _OI __result)`

- `template<typename _Tp >`
`std::remove_reference< _Tp >::type && std::move (_Tp &&__t)`
- `template<typename _BI1 , typename _BI2 >`
`_BI2 std::move_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _ForwardIterator , typename _Predicate >`
`_ForwardIterator std::partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _InputIterator , typename _OutputIterator1 , typename _OutputIterator2 , typename _Predicate >`
`pair< _OutputIterator1, _OutputIterator2 > std::partition_copy (_InputIterator __first, _InputIterator __last, _OutputIterator1 __out_true, _OutputIterator2 __out_false, _Predicate __pred)`
- `template<typename _ForwardIterator , typename _Predicate >`
`_ForwardIterator std::partition_point (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _RandomAccessIterator >`
`void std::random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator , typename _RandomNumberGenerator >`
`void std::random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomNumberGenerator &&__rand)`
- `template<typename _ForwardIterator , typename _Tp >`
`_ForwardIterator std::remove (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _InputIterator , typename _OutputIterator , typename _Tp >`
`_OutputIterator std::remove_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__value)`
- `template<typename _InputIterator , typename _OutputIterator , typename _Predicate >`
`_OutputIterator std::remove_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _ForwardIterator , typename _Predicate >`
`_ForwardIterator std::remove_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator , typename _Tp >`
`void std::replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _InputIterator , typename _OutputIterator , typename _Predicate , typename _Tp >`
`_OutputIterator std::replace_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _ForwardIterator , typename _Predicate , typename _Tp >`
`void std::replace_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _BidirectionalIterator >`
`void std::reverse (_BidirectionalIterator __first, _BidirectionalIterator __last)`

- `template<typename _BidirectionalIterator, typename _OutputIterator >`
`_OutputIterator std::reverse_copy (_BidirectionalIterator __first, _-`
`BidirectionalIterator __last, _OutputIterator __result)`
- `template<typename _ForwardIterator >`
`void std::rotate (_ForwardIterator __first, _ForwardIterator __middle, _-`
`ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _OutputIterator >`
`_OutputIterator std::rotate_copy (_ForwardIterator __first, _ForwardIterator __-`
`middle, _ForwardIterator __last, _OutputIterator __result)`
- `template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator >`
`void std::shuffle (_RandomAccessIterator __first, _RandomAccessIterator __-`
`last, _UniformRandomNumberGenerator && __g)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator std::stable_partition (_ForwardIterator __first, _-`
`ForwardIterator __last, _Predicate __pred)`
- `template<typename _Tp >`
`void std::swap (_Tp &__a, _Tp &__b)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`_ForwardIterator2 std::swap_ranges (_ForwardIterator1 __first1, _-`
`ForwardIterator1 __last1, _ForwardIterator2 __first2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`
`name _BinaryOperation >`
`_OutputIterator std::transform (_InputIterator1 __first1, _InputIterator1 __last1,`
`_InputIterator2 __first2, _OutputIterator __result, _BinaryOperation __binary_-`
`op)`
- `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation >`
`_OutputIterator std::transform (_InputIterator __first, _InputIterator __last, _-`
`OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`
`_ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last,`
`_BinaryPredicate __binary_pred)`
- `template<typename _ForwardIterator >`
`_ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _InputIterator, typename _OutputIterator >`
`_OutputIterator std::unique_copy (_InputIterator __first, _InputIterator __last,`
`_OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate >`
`_OutputIterator std::unique_copy (_InputIterator __first, _InputIterator __last,`
`_OutputIterator __result, _BinaryPredicate __binary_pred)`

2.27.1 Function Documentation

2.27.1.1 `template<typename _II , typename _OI > _OI std::copy (_II __first,
_II __last, _OI __result) [inline]`

Copies the range [first,last) into result.

Parameters

first An input iterator.

last An input iterator.

result An output iterator.

Returns

result + (first - last)

This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling). Result may not be contained within [first,last); the `copy_backward` function should be used instead.

Note that the end of the output range is permitted to be contained within [first,last).

Definition at line 444 of file `stl_algobase.h`.

2.27.1.2 `template<typename _BI1 , typename _BI2 > _BI2
std::copy_backward (_BI1 __first, _BI1 __last, _BI2 __result)
[inline]`

Copies the range [first,last) into result.

Parameters

first A bidirectional iterator.

last A bidirectional iterator.

result A bidirectional iterator.

Returns

result - (first - last)

The function has the same effect as `copy`, but starts at the end of the range and works its way to the start, returning the start of the result. This inline function will boil down

to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Result may not be in the range `[first,last)`. Use `copy` instead. Note that the start of the output range may overlap `[first,last)`.

Definition at line 613 of file `stl_algobase.h`.

2.27.1.3 `template<typename _InputIterator , typename _OutputIterator ,
typename _Predicate > _OutputIterator std::copy_if (_InputIterator
__first, _InputIterator __last, _OutputIterator __result, _Predicate
__pred)`

Copy the elements of a sequence for which a predicate is true.

Parameters

first An input iterator.

last An input iterator.

result An output iterator.

pred A predicate.

Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range `[first,last)` for which `pred` returns true to the range beginning at `result`.

`copy_if()` is stable, so the relative order of elements that are copied is unchanged.

Definition at line 953 of file `stl_algo.h`.

2.27.1.4 `template<typename _InputIterator , typename _Size , typename
_OutputIterator > _OutputIterator std::copy_n (_InputIterator
__first, _Size __n, _OutputIterator __result) [inline]`

Copies the range `[first,first+n)` into `[result,result+n)`.

Parameters

first An input iterator.

n The number of elements to copy.

result An output iterator.

Returns

result+n.

This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Definition at line 1010 of file `stl_algo.h`.

References `std::__iterator_category()`.

```
2.27.1.5  template<typename _ForwardIterator , typename _Tp > void std::fill
           ( _ForwardIterator __first, _ForwardIterator __last, const _Tp &
             __value ) [inline]
```

Fills the range [first,last) with copies of value.

Parameters

first A forward iterator.

last A forward iterator.

value A reference-to-const of arbitrary type.

Returns

Nothing.

This function fills a range with copies of the same value. For char types filling contiguous areas of memory, this becomes an inline call to `memset` or `wmemset`.

Definition at line 715 of file `stl_algobase.h`.

```
2.27.1.6  template<typename _OI , typename _Size , typename _Tp > _OI
           std::fill_n ( _OI __first, _Size __n, const _Tp & __value )
           [inline]
```

Fills the range [first,first+n) with copies of value.

Parameters

- first* An output iterator.
n The count of copies to perform.
value A reference-to-const of arbitrary type.

Returns

The iterator at first+n.

This function fills a range with copies of the same value. For char types filling contiguous areas of memory, this becomes an inline call to `memset` or `@ wmemset`.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 865. More algorithms that throw away information

Definition at line 775 of file `stl_algobase.h`.

2.27.1.7 `template<typename _ForwardIterator , typename _Generator > void
 std::generate (_ForwardIterator __first, _ForwardIterator __last,
 _Generator __gen)`

Assign the result of a function object to each value in a sequence.

Parameters

- first* A forward iterator.
last A forward iterator.
gen A function object taking no arguments and returning `std::iterator_traits<_ForwardIterator>::value_type`

Returns

`generate()` returns no value.

Performs the assignment `*i = gen ()` for each `i` in the range `[first,last)`.

Definition at line 4899 of file `stl_algo.h`.

2.27.1.8 `template<typename _OutputIterator , typename _Size , typename
 _Generator > _OutputIterator std::generate_n (_OutputIterator
 __first, _Size __n, _Generator __gen)`

Assign the result of a function object to each value in a sequence.

Parameters

first A forward iterator.

n The length of the sequence.

gen A function object taking no arguments and returning `std::iterator_traits<_ForwardIterator>::value_type`

Returns

The end of the sequence, `first+n`

Performs the assignment `*i = gen()` for each `i` in the range `[first,first+n)`.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 865. More algorithms that throw away information

Definition at line 4930 of file `stl_algo.h`.

2.27.1.9 `template<typename _InputIterator , typename _Predicate > bool
std::is_partitioned (_InputIterator __first, _InputIterator __last,
_Predicate __pred) [inline]`

Checks whether the sequence is partitioned.

Parameters

first An input iterator.

last An input iterator.

pred A predicate.

Returns

True if the range `[first,last)` is partitioned by `pred`, i.e. if all elements that satisfy `pred` appear before those that do not.

Definition at line 805 of file `stl_algo.h`.

References `std::find_if_not()`, and `std::none_of()`.

2.27.1.10 `template<typename _ForwardIterator1 , typename
_ForwardIterator2 > void std::iter_swap (_ForwardIterator1 __a,
_ForwardIterator2 __b) [inline]`

Swaps the contents of two iterators.

Parameters

- a* An iterator.
- b* Another iterator.

Returns

Nothing.

This function swaps the values pointed to by two iterators, not the iterators themselves.

Definition at line 118 of file `stl_algobase.h`.

Referenced by `std::__merge_without_buffer()`, `std::__move_median_first()`, `std::__partition()`, `std::__reverse()`, `std::__rotate()`, `std::__unguarded_partition()`, `std::next_permutation()`, `std::prev_permutation()`, `std::random_shuffle()`, `std::shuffle()`, and `std::swap_ranges()`.

2.27.1.11 `template<typename _II, typename _OI> _OI std::move (_II
_first, _II _last, _OI _result) [inline]`

Moves the range `[first,last)` into `result`.

Parameters

- first* An input iterator.
- last* An input iterator.
- result* An output iterator.

Returns

`result + (first - last)`

This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling). Result may not be contained within `[first,last)`; the `move_backward` function should be used instead.

Note that the end of the output range is permitted to be contained within `[first,last)`.

Definition at line 477 of file `stl_algobase.h`.

2.27.1.12 `template<typename _Tp> std::remove_reference<_Tp>::type&&
std::move (_Tp && __t) [inline]`

Move a value.

Parameters

__t A thing of arbitrary type.

Returns

Same, moved.

Definition at line 82 of file move.h.

2.27.1.13 `template<typename _BI1 , typename _BI2 > _BI2
std::move_backward (_BI1 __first, _BI1 __last, _BI2 __result)
[inline]`

Moves the range [first,last) into result.

Parameters

first A bidirectional iterator.

last A bidirectional iterator.

result A bidirectional iterator.

Returns

result - (first - last)

The function has the same effect as move, but starts at the end of the range and works its way to the start, returning the start of the result. This inline function will boil down to a call to memmove whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Result may not be in the range [first,last). Use move instead. Note that the start of the output range may overlap [first,last).

Definition at line 649 of file stl_algobase.h.

2.27.1.14 `template<typename _ForwardIterator , typename _Predicate
> _ForwardIterator std::partition (_ForwardIterator __first,
_ForwardIterator __last, _Predicate __pred) [inline]`

Move elements for which a predicate is true to the beginning of a sequence.

Parameters

first A forward iterator.
last A forward iterator.
pred A predicate functor.

Returns

An iterator `middle` such that `pred(i)` is true for each iterator `i` in the range `[first,middle)` and false for each `i` in the range `[middle,last)`.

`pred` must not modify its operand. `partition()` does not preserve the relative ordering of elements in each group, use `stable_partition()` if this is needed.

Definition at line 5102 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__partition()`.

2.27.1.15 `template<typename _InputIterator , typename _OutputIterator1
, typename _OutputIterator2 , typename _Predicate >
pair<_OutputIterator1, _OutputIterator2> std::partition_copy (
_InputIterator __first, _InputIterator __last, _OutputIterator1
__out_true, _OutputIterator2 __out_false, _Predicate __pred)`

Copy the elements of a sequence to separate output sequences depending on the truth value of a predicate.

Parameters

first An input iterator.
last An input iterator.
out_true An output iterator.
out_false An output iterator.
pred A predicate.

Returns

A pair designating the ends of the resulting sequences.

Copies each element in the range `[first,last)` for which `pred` returns true to the range beginning at `out_true` and each element for which `pred` returns false to `out_false`.

Definition at line 1039 of file `stl_algo.h`.

2.27.1.16 `template<typename _ForwardIterator, typename _Predicate >
_ForwardIterator std::partition_point (_ForwardIterator __first,
_ForwardIterator __last, _Predicate __pred)`

Find the partition point of a partitioned range.

Parameters

first An iterator.

last Another iterator.

pred A predicate.

Returns

An iterator *mid* such that `all_of(first, mid, pred)` and `none_of(mid, last, pred)` are both true.

Definition at line 823 of file `stl_algo.h`.

References `std::advance()`, and `std::distance()`.

2.27.1.17 `template<typename _RandomAccessIterator > void
std::random_shuffle (_RandomAccessIterator __first,
_RandomAccessIterator __last) [inline]`

Randomly shuffle the elements of a sequence.

Parameters

first A forward iterator.

last A forward iterator.

Returns

Nothing.

Reorder the elements in the range `[first,last)` using a random distribution, so that every possible ordering of the sequence is equally likely.

Definition at line 5038 of file `stl_algo.h`.

References `std::iter_swap()`.

2.27.1.18 `template<typename _RandomAccessIterator, typename
_RandomNumberGenerator > void std::random_shuffle (
_RandomAccessIterator __first, _RandomAccessIterator __last,
_RandomNumberGenerator && __rand)`

Shuffle the elements of a sequence using a random number generator.

Parameters

first A forward iterator.

last A forward iterator.

rand The RNG functor or function.

Returns

Nothing.

Reorders the elements in the range [first,last) using *rand* to provide a random distribution. Calling *rand*(N) for a positive integer N should return a randomly chosen integer from the range [0,N).

Definition at line 5066 of file stl_algo.h.

References std::iter_swap().

2.27.1.19 `template<typename _ForwardIterator, typename _Tp >
_ForwardIterator std::remove (_ForwardIterator __first,
_ForwardIterator __last, const _Tp & __value)`

Remove elements from a sequence.

Parameters

first An input iterator.

last An input iterator.

value The value to be removed.

Returns

An iterator designating the end of the resulting sequence.

All elements equal to *value* are removed from the range [first,last).

remove() is stable, so the relative order of elements that are not removed is unchanged.

Elements between the end of the resulting sequence and `last` are still present, but their value is unspecified.

Definition at line 1088 of file `stl_algo.h`.

2.27.1.20 `template<typename _InputIterator, typename _OutputIterator, typename _Tp> _OutputIterator std::remove_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp & __value)`

Copy a sequence, removing elements of a given value.

Parameters

first An input iterator.
last An input iterator.
result An output iterator.
value The value to be removed.

Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range `[first,last)` not equal to `value` to the range beginning at `result`. `remove_copy()` is stable, so the relative order of elements that are copied is unchanged.

Definition at line 876 of file `stl_algo.h`.

2.27.1.21 `template<typename _InputIterator, typename _OutputIterator, typename _Predicate> _OutputIterator std::remove_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`

Copy a sequence, removing elements for which a predicate is true.

Parameters

first An input iterator.
last An input iterator.
result An output iterator.
pred A predicate.

Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range `[first,last)` for which `pred` returns false to the range beginning at `result`.

`remove_copy_if()` is stable, so the relative order of elements that are copied is unchanged.

Definition at line 914 of file `stl_algo.h`.

```
2.27.1.22  template<typename _ForwardIterator , typename _Predicate >
             _ForwardIterator std::remove_if ( _ForwardIterator __first,
             _ForwardIterator __last, _Predicate __pred )
```

Remove elements from a sequence using a predicate.

Parameters

first A forward iterator.

last A forward iterator.

pred A predicate.

Returns

An iterator designating the end of the resulting sequence.

All elements for which `pred` returns true are removed from the range `[first,last)`.

`remove_if()` is stable, so the relative order of elements that are not removed is unchanged.

Elements between the end of the resulting sequence and `last` are still present, but their value is unspecified.

Definition at line 1131 of file `stl_algo.h`.

```
2.27.1.23  template<typename _ForwardIterator , typename _Tp > void
             std::replace ( _ForwardIterator __first, _ForwardIterator __last,
             const _Tp & __old_value, const _Tp & __new_value )
```

Replace each occurrence of one value in a sequence with another value.

Parameters

first A forward iterator.
last A forward iterator.
old_value The value to be replaced.
new_value The replacement value.

Returns

replace() returns no value.

For each iterator *i* in the range [first,last) if **i == old_value* then the assignment **i = new_value* is performed.

Definition at line 4835 of file stl_algo.h.

2.27.1.24 `template<typename _InputIterator, typename _OutputIterator
, typename _Predicate, typename _Tp > _OutputIterator
std::replace_copy_if (_InputIterator __first, _InputIterator
__last, _OutputIterator __result, _Predicate __pred, const _Tp &
__new_value)`

Copy a sequence, replacing each value for which a predicate returns true with another value.

Parameters

first An input iterator.
last An input iterator.
result An output iterator.
pred A predicate.
new_value The replacement value.

Returns

The end of the output sequence, *result+(last-first)*.

Copies each element in the range [first,last) to the range [result,result+(last-first)) replacing elements for which *pred* returns true with *new_value*.

Definition at line 3783 of file stl_algo.h.

2.27.1.25 `template<typename _ForwardIterator, typename _Predicate, typename _Tp> void std::replace_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, const _Tp & __new_value)`

Replace each value in a sequence for which a predicate returns true with another value.

Parameters

first A forward iterator.

last A forward iterator.

pred A predicate.

new_value The replacement value.

Returns

`replace_if()` returns no value.

For each iterator *i* in the range `[first,last)` if `pred(*i)` is true then the assignment `*i = new_value` is performed.

Definition at line 4867 of file `stl_algo.h`.

2.27.1.26 `template<typename _BidirectionalIterator> void std::reverse (_BidirectionalIterator __first, _BidirectionalIterator __last) [inline]`

Reverse a sequence.

Parameters

first A bidirectional iterator.

last A bidirectional iterator.

Returns

`reverse()` returns no value.

Reverses the order of the elements in the range `[first,last)`, so that the first element becomes the last etc. For every *i* such that $0 \leq i \leq (\text{last} - \text{first})/2$, `reverse()` swaps `*(first+i)` and `*(last-(i+1))`.

Definition at line 1439 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__reverse()`.

Referenced by `std::next_permutation()`, and `std::prev_permutation()`.

2.27.1.27 `template<typename _BidirectionalIterator , typename
_OutputIterator > _OutputIterator std::reverse_copy (
_BidirectionalIterator __first, _BidirectionalIterator __last,
_OutputIterator __result)`

Copy a sequence, reversing its elements.

Parameters

first A bidirectional iterator.

last A bidirectional iterator.

result An output iterator.

Returns

An iterator designating the end of the resulting sequence.

Copies the elements in the range `[first,last)` to the range `[result,result+(last-first))` such that the order of the elements is reversed. For every `i` such that $0 \leq i < (last - first)$, `reverse_copy()` performs the assignment `*(result+(last-first)-i) = *(first+i)`. The ranges `[first,last)` and `[result,result+(last-first))` must not overlap.

Definition at line 1466 of file `stl_algo.h`.

2.27.1.28 `template<typename _ForwardIterator > void std::rotate
(_ForwardIterator __first, _ForwardIterator __middle,
_ForwardIterator __last) [inline]`

Rotate the elements of a sequence.

Parameters

first A forward iterator.

middle A forward iterator.

last A forward iterator.

Returns

Nothing.

Rotates the elements of the range `[first,last)` by `(middle-first)` positions so that the element at `middle` is moved to `first`, the element at `middle+1` is moved to `+1` and so on for each element in the range `[first,last)`.

This effectively swaps the ranges `[first,middle)` and `[middle,last)`.

Performs $*(first+(n+(last-middle))\%(last-first))=*(first+n)$ for each `n` in the range `[0,last-first)`.

Definition at line 1670 of file `stl_algo.h`.

References `std::__rotate()`.

Referenced by `std::__inplace_stable_partition()`, `std::__merge_without_buffer()`, `std::__rotate_adaptive()`, and `std::__stable_partition_adaptive()`.

2.27.1.29 `template<typename _ForwardIterator, typename _OutputIterator
> _OutputIterator std::rotate_copy (_ForwardIterator
__first, _ForwardIterator __middle, _ForwardIterator __last,
_OutputIterator __result)`

Copy a sequence, rotating its elements.

Parameters

first A forward iterator.

middle A forward iterator.

last A forward iterator.

result An output iterator.

Returns

An iterator designating the end of the resulting sequence.

Copies the elements of the range `[first,last)` to the range beginning at

Returns

, rotating the copied elements by `(middle-first)` positions so that the element at `middle` is moved to `result`, the element at `middle+1` is moved to `+1` and so on for each element in the range `[first,last)`.

Performs $*(result+(n+(last-middle))\%(last-first))=*(first+n)$ for each `n` in the range `[0,last-first)`.

Definition at line 1704 of file `stl_algo.h`.

2.27.1.30 `template<typename _RandomAccessIterator, typename
_UniformRandomNumberGenerator > void std::shuffle (`
 `_RandomAccessIterator __first, _RandomAccessIterator __last,`
 `_UniformRandomNumberGenerator && __g)`

Shuffle the elements of a sequence using a uniform random number generator.

Parameters

first A forward iterator.

last A forward iterator.

g A UniformRandomNumberGenerator (26.5.1.3).

Returns

Nothing.

Reorders the elements in the range [first,last) using *g* to provide random numbers.

Definition at line 4225 of file stl_algo.h.

References std::iter_swap().

2.27.1.31 `template<typename _ForwardIterator, typename _Predicate >`
 `_ForwardIterator std::stable_partition (_ForwardIterator __first,`
 `_ForwardIterator __last, _Predicate __pred)`

Move elements for which a predicate is true to the beginning of a sequence, preserving relative ordering.

Parameters

first A forward iterator.

last A forward iterator.

pred A predicate functor.

Returns

An iterator `middle` such that `pred(i)` is true for each iterator *i* in the range [first,middle) and false for each *i* in the range [middle,last).

Performs the same function as `partition()` with the additional guarantee that the relative ordering of elements in each group is preserved, so any two elements *x* and

y in the range $[first, last)$ such that $pred(x) == pred(y)$ will have the same relative ordering after calling `stable_partition()`.

Definition at line 1862 of file `stl_algo.h`.

References `std::__inplace_stable_partition()`, `std::__stable_partition_adaptive()`, `std::_Temporary_buffer<_ForwardIterator, _Tp>::begin()`, `std::_Temporary_buffer<_ForwardIterator, _Tp>::requested_size()`, and `std::_Temporary_buffer<_ForwardIterator, _Tp>::size()`.

2.27.1.32 `template<typename _Tp> void std::swap (_Tp & __a, _Tp & __b) [inline]`

Swaps two values.

Parameters

- `__a` A thing of arbitrary type.
- `__b` Another thing of arbitrary type.

Returns

Nothing.

Definition at line 122 of file `move.h`.

2.27.1.33 `template<typename _ForwardIterator1, typename _ForwardIterator2> _ForwardIterator2 std::swap_ranges (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`

Swap the elements of two sequences.

Parameters

- first1* A forward iterator.
- last1* A forward iterator.
- first2* A forward iterator.

Returns

An iterator equal to `first2+(last1-first1)`.

Swaps each element in the range `[first1,last1)` with the corresponding element in the range `[first2,(last1-first1))`. The ranges must not overlap.

Definition at line 159 of file `stl_algo.h`.

References `std::iter_swap()`.

Referenced by `std::__rotate()`.

2.27.1.34 `template<typename _InputIterator1, typename _InputIterator2
, typename _OutputIterator, typename _BinaryOperation
> _OutputIterator std::transform (_InputIterator1 __first1,
_InputIterator1 __last1, _InputIterator2 __first2, _OutputIterator
__result, _BinaryOperation __binary_op)`

Perform an operation on corresponding elements of two sequences.

Parameters

first1 An input iterator.

last1 An input iterator.

first2 An input iterator.

result An output iterator.

binary_op A binary operator.

Returns

An output iterator equal to `result+(last-first)`.

Applies the operator to the corresponding elements in the two input ranges and assigns the results to successive elements of the output sequence. Evaluates `*(result+N)=binary_op(*(first1+N),*(first2+N))` for each `N` in the range `[0,last1-first1)`.

`binary_op` must not alter either of its arguments.

Definition at line 4803 of file `stl_algo.h`.

2.27.1.35 `template<typename _InputIterator, typename _OutputIterator,
typename _UnaryOperation > _OutputIterator std::transform (_InputIterator
__first, _InputIterator __last, _OutputIterator
__result, _UnaryOperation __unary_op)`

Perform an operation on a sequence.

Parameters

first An input iterator.
last An input iterator.
result An output iterator.
unary_op A unary operator.

Returns

An output iterator equal to `result+(last-first)`.

Applies the operator to each element in the input range and assigns the results to successive elements of the output sequence. Evaluates `*(result+N)=unary_op(*(first+N))` for each `N` in the range `[0,last-first)`.

`unary_op` must not alter its argument.

Definition at line 4767 of file `stl_algo.h`.

2.27.1.36 `template<typename _ForwardIterator, typename _BinaryPredicate
 > _ForwardIterator std::unique (_ForwardIterator __first,
 _ForwardIterator __last, _BinaryPredicate __binary_pred)`

Remove consecutive values from a sequence using a predicate.

Parameters

first A forward iterator.
last A forward iterator.
binary_pred A binary predicate.

Returns

An iterator designating the end of the resulting sequence.

Removes all but the first element from each group of consecutive values for which `binary_pred` returns true. `unique()` is stable, so the relative order of elements that are not removed is unchanged. Elements between the end of the resulting sequence and `last` are still present, but their value is unspecified.

Definition at line 1211 of file `stl_algo.h`.

2.27.1.37 `template<typename _ForwardIterator > _ForwardIterator
 std::unique (_ForwardIterator __first, _ForwardIterator __last)`

Remove consecutive duplicate values from a sequence.

Parameters

first A forward iterator.

last A forward iterator.

Returns

An iterator designating the end of the resulting sequence.

Removes all but the first element from each group of consecutive values that compare equal. `unique()` is stable, so the relative order of elements that are not removed is unchanged. Elements between the end of the resulting sequence and `last` are still present, but their value is unspecified.

Definition at line 1171 of file `stl_algo.h`.

2.27.1.38 `template<typename _InputIterator, typename _OutputIterator
> _OutputIterator std::unique_copy (_InputIterator __first,
_InputIterator __last, _OutputIterator __result) [inline]`

Copy a sequence, removing consecutive duplicate values.

Parameters

first An input iterator.

last An input iterator.

result An output iterator.

Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range `[first,last)` to the range beginning at `result`, except that only the first element is copied from groups of consecutive elements that compare equal. `unique_copy()` is stable, so the relative order of elements that are copied is unchanged.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 241. Does `unique_copy()` require Copy-Constructible and Assignable?

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 538. 241 again: Does `unique_copy()` require CopyConstructible and Assignable?

Definition at line 4967 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__unique_copy()`.

2.27.1.39 `template<typename _InputIterator, typename _OutputIterator,
typename _BinaryPredicate > _OutputIterator std::unique_copy(
_InputIterator __first, _InputIterator __last, _OutputIterator
__result, _BinaryPredicate __binary_pred) [inline]`

Copy a sequence, removing consecutive values using a predicate.

Parameters

first An input iterator.
last An input iterator.
result An output iterator.
binary_pred A binary predicate.

Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range `[first,last)` to the range beginning at `result`, except that only the first element is copied from groups of consecutive elements for which `binary_pred` returns true. `unique_copy()` is stable, so the relative order of elements that are copied is unchanged.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 241. Does `unique_copy()` require Copy-Constructible and Assignable?

Definition at line 5007 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__unique_copy()`.

2.28 Non-Mutating

Collaboration diagram for Non-Mutating:



Functions

- `template<typename _ForwardIterator >`
`_ForwardIterator std::adjacent_find (_ForwardIterator __first, _ForwardIterator`
`__last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`
`_ForwardIterator std::adjacent_find (_ForwardIterator __first, _ForwardIterator`
`__last, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Predicate >`
`bool std::all_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate >`
`bool std::any_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Tp >`
`iterator_traits< _InputIterator >::difference_type std::count (_InputIterator __-`
`first, _InputIterator __last, const _Tp &__value)`
- `template<typename _InputIterator, typename _Predicate >`
`iterator_traits< _InputIterator >::difference_type std::count_if (_InputIterator`
`__first, _InputIterator __last, _Predicate __pred)`
- `template<typename _II1, typename _II2 >`
`bool std::equal (_II1 __first1, _II1 __last1, _II2 __first2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`
`bool std::equal (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _-`
`BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Tp >`
`_InputIterator std::find (_InputIterator __first, _InputIterator __last, const _Tp`
`&__val)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate`
`>`
`_ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _ForwardIterator1`
`__last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _-`
`BinaryPredicate __comp)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`_ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _ForwardIterator1`
`__last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _InputIterator, typename _ForwardIterator >`
`_InputIterator std::find_first_of (_InputIterator __first1, _InputIterator __last1,`
`_ForwardIterator __first2, _ForwardIterator __last2)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`
`_InputIterator std::find_first_of (_InputIterator __first1, _InputIterator __last1,`
`_ForwardIterator __first2, _ForwardIterator __last2, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _Predicate >`
`_InputIterator std::find_if (_InputIterator __first, _InputIterator __last, _-`
`Predicate __pred)`

- `template<typename _InputIterator, typename _Predicate >`
`_InputIterator std::find_if_not (_InputIterator __first, _InputIterator __last, _`
`Predicate __pred)`
- `template<typename _InputIterator, typename _Function >`
`_Function std::for_each (_InputIterator __first, _InputIterator __last, _Function`
`__f)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate`
`>`
`bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1,`
`_ForwardIterator2 __first2, _BinaryPredicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1,`
`_ForwardIterator2 __first2)`
- `template<typename _InputIterator1, typename _InputIterator2 >`
`pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1,`
`_InputIterator1 __last1, _InputIterator2 __first2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`
`pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1,`
`_InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_`
`pred)`
- `template<typename _InputIterator, typename _Predicate >`
`bool std::none_of (_InputIterator __first, _InputIterator __last, _Predicate __`
`pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`_ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 _`
`__last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate`
`>`
`_ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 _`
`__last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate`
`__predicate)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _`
`BinaryPredicate >`
`_ForwardIterator std::search_n (_ForwardIterator __first, _ForwardIterator __`
`last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp >`
`_ForwardIterator std::search_n (_ForwardIterator __first, _ForwardIterator __`
`last, _Integer __count, const _Tp &__val)`

2.28.1 Function Documentation

2.28.1.1 `template<typename _ForwardIterator > _ForwardIterator
std::adjacent_find (_ForwardIterator __first, _ForwardIterator
__last)`

Find two adjacent values in a sequence that are equal.

Parameters

first A forward iterator.

last A forward iterator.

Returns

The first iterator *i* such that *i* and *i*+1 are both valid iterators in [*first*,*last*) and such that `*i == *(i+1)`, or *last* if no such iterator exists.

Definition at line 4414 of file `stl_algo.h`.

2.28.1.2 `template<typename _ForwardIterator , typename _BinaryPredicate >
_ForwardIterator std::adjacent_find (_ForwardIterator __first,
_ForwardIterator __last, _BinaryPredicate __binary_pred)`

Find two adjacent values in a sequence using a predicate.

Parameters

first A forward iterator.

last A forward iterator.

binary_pred A binary predicate.

Returns

The first iterator *i* such that *i* and *i*+1 are both valid iterators in [*first*,*last*) and such that `binary_pred(*i,*i+1)` is true, or *last* if no such iterator exists.

Definition at line 4446 of file `stl_algo.h`.

2.28.1.3 `template<typename _InputIterator , typename _Predicate > bool
std::all_of (_InputIterator __first, _InputIterator __last, _Predicate
__pred) [inline]`

Checks that a predicate is true for all the elements of a sequence.

Parameters

first An input iterator.

last An input iterator.

pred A predicate.

Returns

True if the check is true, false otherwise.

Returns true if `pred` is true for each element in the range `[first,last)`, and false otherwise.

Definition at line 732 of file `stl_algo.h`.

References `std::find_if_not()`.

```
2.28.1.4 template<typename _InputIterator , typename _Predicate >
        bool std::any_of ( _InputIterator __first, _InputIterator __last,
                          _Predicate __pred ) [inline]
```

Checks that a predicate is false for at least an element of a sequence.

Parameters

first An input iterator.

last An input iterator.

pred A predicate.

Returns

True if the check is true, false otherwise.

Returns true if an element exists in the range `[first,last)` such that `pred` is true, and false otherwise.

Definition at line 766 of file `stl_algo.h`.

References `std::none_of()`.

```
2.28.1.5 template<typename _InputIterator , typename _Tp >
        iterator_traits<_InputIterator>::difference_type std::count (
        _InputIterator __first, _InputIterator __last, const _Tp & __value )
```


Count the number of copies of a value in a sequence.

Parameters

first An input iterator.
last An input iterator.
value The value to be counted.

Returns

The number of iterators *i* in the range [first,last) for which `*i == value`

Definition at line 4478 of file `stl_algo.h`.

Referenced by `std::is_permutation()`.

2.28.1.6 `template<typename _InputIterator , typename _Predicate >
 iterator_traits<_InputIterator>::difference_type std::count_if (
 _InputIterator __first, _InputIterator __last, _Predicate __pred)`

Count the elements of a sequence for which a predicate is true.

Parameters

first An input iterator.
last An input iterator.
pred A predicate.

Returns

The number of iterators *i* in the range [first,last) for which `pred(*i)` is true.

Definition at line 4503 of file `stl_algo.h`.

Referenced by `std::is_permutation()`.

2.28.1.7 `template<typename _II1 , typename _II2 > bool std::equal (_II1
 __first1, _II1 __last1, _II2 __first2) [inline]`

Tests a range for element-wise equality.

Parameters

first1 An input iterator.

last1 An input iterator.

first2 An input iterator.

Returns

A boolean true or false.

This compares the elements of two ranges using == and returns true or false depending on whether all of the corresponding elements of the ranges are equal.

Definition at line 1010 of file stl_algobase.h.

Referenced by std::operator==().

2.28.1.8 `template<typename _Iter1 , typename _Iter2 , typename
_BinaryPredicate > bool std::equal (_Iter1 __first1, _Iter1 __last1,
_Iter2 __first2, _BinaryPredicate __binary_pred) [inline]`

Tests a range for element-wise equality.

Parameters

first1 An input iterator.

last1 An input iterator.

first2 An input iterator.

binary_pred A binary predicate [functor](#).

Returns

A boolean true or false.

This compares the elements of two ranges using the `binary_pred` parameter, and returns true or false depending on whether all of the corresponding elements of the ranges are equal.

Definition at line 1042 of file stl_algobase.h.

2.28.1.9 `template<typename _InputIterator , typename _Tp > _InputIterator
std::find (_InputIterator __first, _InputIterator __last, const _Tp &
__val) [inline]`

Find the first occurrence of a value in a sequence.

Parameters

first An input iterator.

last An input iterator.

val The value to find.

Returns

The first iterator *i* in the range `[first,last)` such that `*i == val`, or `last` if no such iterator exists.

Definition at line 4290 of file `stl_algo.h`.

References `std::__find()`, and `std::__iterator_category()`.

```
2.28.1.10 template<typename _ForwardIterator1 , typename
               _ForwardIterator2 , typename _BinaryPredicate >
               _ForwardIterator1 std::find_end ( _ForwardIterator1 __first1,
               _ForwardIterator1 __last1, _ForwardIterator2 __first2,
               _ForwardIterator2 __last2, _BinaryPredicate __comp )
               [inline]
```

Find last matching subsequence in a sequence using a predicate.

Parameters

first1 Start of range to search.

last1 End of range to search.

first2 Start of sequence to match.

last2 End of sequence to match.

comp The predicate to use.

Returns

The last iterator *i* in the range `[first1,last1-(last2-first2))` such that `predicate(*(i+N), (first2+N))` is true for each *N* in the range `[0,last2-first2)`, or `last1` if no such iterator exists.

Searches the range `[first1,last1)` for a sub-sequence that compares equal value-by-value with the sequence given by `[first2,last2)` using `comp` as a predicate and returns an iterator to the first element of the sub-sequence, or `last1` if the sub-sequence is not found. The sub-sequence will be the last such subsequence contained in `[first,last1)`.

Because the sub-sequence must lie completely within the range `[first1,last1)` it must start at a position less than `last1-(last2-first2)` where `last2-first2` is the length

of the sub-sequence. This means that the returned iterator `i` will be in the range `[first1, last1 - (last2 - first2))`

Definition at line 698 of file `stl_algo.h`.

References `std::__iterator_category()`.

```
2.28.1.11  template<typename _ForwardIterator1 , typename
              _ForwardIterator2 > _ForwardIterator1 std::find_end (
              _ForwardIterator1 __first1, _ForwardIterator1 __last1,
              _ForwardIterator2 __first2, _ForwardIterator2 __last2 )
              [inline]
```

Find last matching subsequence in a sequence.

Parameters

first1 Start of range to search.

last1 End of range to search.

first2 Start of sequence to match.

last2 End of sequence to match.

Returns

The last iterator `i` in the range `[first1, last1 - (last2 - first2))` such that `*(i+N) == *(first2+N)` for each `N` in the range `[0, last2 - first2)`, or `last1` if no such iterator exists.

Searches the range `[first1, last1)` for a sub-sequence that compares equal value-by-value with the sequence given by `[first2, last2)` and returns an iterator to the first element of the sub-sequence, or `last1` if the sub-sequence is not found. The sub-sequence will be the last such subsequence contained in `[first, last1)`.

Because the sub-sequence must lie completely within the range `[first1, last1)` it must start at a position less than `last1 - (last2 - first2)` where `last2 - first2` is the length of the sub-sequence. This means that the returned iterator `i` will be in the range `[first1, last1 - (last2 - first2))`

Definition at line 651 of file `stl_algo.h`.

References `std::__iterator_category()`.

2.28.1.12 `template<typename _InputIterator, typename _ForwardIterator
> _InputIterator std::find_first_of (_InputIterator
__first1, _InputIterator __last1, _ForwardIterator __first2,
_ForwardIterator __last2)`

Find element from a set in a sequence.

Parameters

first1 Start of range to search.

last1 End of range to search.

first2 Start of match candidates.

last2 End of match candidates.

Returns

The first iterator *i* in the range [*first1*,*last1*) such that **i* == **(i2)* such that *i2* is an iterator in [*first2*,*last2*), or *last1* if no such iterator exists.

Searches the range [*first1*,*last1*) for an element that is equal to some element in the range [*first2*,*last2*). If found, returns an iterator in the range [*first1*,*last1*), otherwise returns *last1*.

Definition at line 4343 of file `stl_algo.h`.

2.28.1.13 `template<typename _InputIterator, typename _ForwardIterator ,
typename _BinaryPredicate > _InputIterator std::find_first_of (
_InputIterator __first1, _InputIterator __last1, _ForwardIterator
__first2, _ForwardIterator __last2, _BinaryPredicate __comp)`

Find element from a set in a sequence using a predicate.

Parameters

first1 Start of range to search.

last1 End of range to search.

first2 Start of match candidates.

last2 End of match candidates.

comp Predicate to use.

Returns

The first iterator *i* in the range [*first1*,*last1*) such that *comp(*i, *(i2))* is true and *i2* is an iterator in [*first2*,*last2*), or *last1* if no such iterator exists.

Searches the range `[first1,last1)` for an element that is equal to some element in the range `[first2,last2)`. If found, returns an iterator in the range `[first1,last1)`, otherwise returns `last1`.

Definition at line 4383 of file `stl_algo.h`.

2.28.1.14 `template<typename _InputIterator, typename _Predicate >
_InputIterator std::find_if (_InputIterator __first, _InputIterator
__last, _Predicate __pred) [inline]`

Find the first element in a sequence for which a predicate is true.

Parameters

first An input iterator.

last An input iterator.

pred A predicate.

Returns

The first iterator `i` in the range `[first,last)` such that `pred(*i)` is true, or `last` if no such iterator exists.

Definition at line 4314 of file `stl_algo.h`.

References `std::__find_if()`, and `std::__iterator_category()`.

2.28.1.15 `template<typename _InputIterator, typename _Predicate
> _InputIterator std::find_if_not (_InputIterator __first,
_InputIterator __last, _Predicate __pred) [inline]`

Find the first element in a sequence for which a predicate is false.

Parameters

first An input iterator.

last An input iterator.

pred A predicate.

Returns

The first iterator `i` in the range `[first,last)` such that `pred(*i)` is false, or `last` if no such iterator exists.

Definition at line 781 of file `stl_algo.h`.

References `std::__find_if_not()`, and `std::__iterator_category()`.

Referenced by `std::all_of()`, and `std::is_partitioned()`.

2.28.1.16 `template<typename _InputIterator, typename _Function >
_Function std::for_each (_InputIterator __first, _InputIterator
__last, _Function __f)`

Apply a function to every element of a sequence.

Parameters

first An input iterator.

last An input iterator.

f A unary function object.

Returns

f (`std::move(f)` in C++0x).

Applies the function object *f* to each element in the range `[first,last)`. *f* must not modify the order of the sequence. If *f* has a return value it is ignored.

Definition at line 4269 of file `stl_algo.h`.

2.28.1.17 `template<typename _ForwardIterator1, typename
_ForwardIterator2, typename _BinaryPredicate >
bool std::is_permutation (_ForwardIterator1 __first1,
_ForwardIterator1 __last1, _ForwardIterator2 __first2,
_BinaryPredicate __pred)`

Checks whether a permutation of the second sequence is equal to the first sequence.

Parameters

first1 Start of first range.

last1 End of first range.

first2 Start of second range.

pred A binary predicate.

Returns

true if there exists a permutation of the elements in the range [first2, first2 + (last1 - first1)), beginning with ForwardIterator2 begin, such that equal(first1, last1, begin, pred) returns true; otherwise, returns false.

Definition at line 4175 of file stl_algo.h.

References std::advance(), std::bind(), std::count_if(), and std::distance().

2.28.1.18 `template<typename _ForwardIterator1 , typename
_ForwardIterator2 > bool std::is_permutation (_ForwardIterator1
__first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`

Checks whether a permutaion of the second sequence is equal to the first sequence.

Parameters

first1 Start of first range.

last1 End of first range.

first2 Start of second range.

Returns

true if there exists a permutation of the elements in the range [first2, first2 + (last1 - first1)), beginning with ForwardIterator2 begin, such that equal(first1, last1, begin) returns true; otherwise, returns false.

Definition at line 4130 of file stl_algo.h.

References std::advance(), std::count(), and std::distance().

2.28.1.19 `template<typename _InputIterator1 , typename _InputIterator2
> pair<_InputIterator1, _InputIterator2> std::mismatch (
_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2
__first2)`

Finds the places in ranges which don't match.

Parameters

first1 An input iterator.

last1 An input iterator.

first2 An input iterator.

Returns

A pair of iterators pointing to the first mismatch.

This compares the elements of two ranges using == and returns a pair of iterators. The first iterator points into the first range, the second iterator points into the second range, and the elements pointed to by the iterators are not equal.

Definition at line 1147 of file stl_algobase.h.

```
2.28.1.20  template<typename _InputIterator1 , typename _InputIterator2
            , typename _BinaryPredicate > pair<_InputIterator1,
            _InputIterator2> std::mismatch ( _InputIterator1 __first1,
            _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate
            __binary_pred )
```

Finds the places in ranges which don't match.

Parameters

first1 An input iterator.

last1 An input iterator.

first2 An input iterator.

binary_pred A binary predicate [functor](#).

Returns

A pair of iterators pointing to the first mismatch.

This compares the elements of two ranges using the *binary_pred* parameter, and returns a pair of iterators. The first iterator points into the first range, the second iterator points into the second range, and the elements pointed to by the iterators are not equal.

Definition at line 1185 of file stl_algobase.h.

```
2.28.1.21  template<typename _InputIterator , typename _Predicate > bool
            std::none_of ( _InputIterator __first, _InputIterator __last,
            _Predicate __pred ) [inline]
```

Checks that a predicate is false for all the elements of a sequence.

Parameters

first An input iterator.

last An input iterator.

pred A predicate.

Returns

True if the check is true, false otherwise.

Returns true if *pred* is false for each element in the range `[first,last)`, and false otherwise.

Definition at line 749 of file `stl_algo.h`.

Referenced by `std::any_of()`, and `std::is_partitioned()`.

2.28.1.22 `template<typename _ForwardIterator1 , typename
_ForwardIterator2 > _ForwardIterator1 std::search (
_ForwardIterator1 __first1, _ForwardIterator1 __last1,
_ForwardIterator2 __first2, _ForwardIterator2 __last2)`

Search a sequence for a matching sub-sequence.

Parameters

first1 A forward iterator.

last1 A forward iterator.

first2 A forward iterator.

last2 A forward iterator.

Returns

The first iterator *i* in the range `[first1,last1-(last2-first2))` such that `*(i+N) == *(first2+N)` for each *N* in the range `[0,last2-first2)`, or *last1* if no such iterator exists.

Searches the range `[first1,last1)` for a sub-sequence that compares equal value-by-value with the sequence given by `[first2,last2)` and returns an iterator to the first element of the sub-sequence, or *last1* if the sub-sequence is not found.

Because the sub-sequence must lie completely within the range `[first1,last1)` it must start at a position less than `last1-(last2-first2)` where `last2-first2` is the length of the sub-sequence. This means that the returned iterator *i* will be in the range `[first1,last1-(last2-first2))`

Definition at line 4543 of file `stl_algo.h`.

2.28.1.23 `template<typename _ForwardIterator1 , typename
_ForwardIterator2 , typename _BinaryPredicate >
_ForwardIterator1 std::search (_ForwardIterator1 __first1,
_ForwardIterator1 __last1, _ForwardIterator2 __first2,
_ForwardIterator2 __last2, _BinaryPredicate __predicate)`

Search a sequence for a matching sub-sequence using a predicate.

Parameters

first1 A forward iterator.
last1 A forward iterator.
first2 A forward iterator.
last2 A forward iterator.
predicate A binary predicate.

Returns

The first iterator *i* in the range `[first1,last1-(last2-first2))` such that `predicate(*(i+N),*(first2+N))` is true for each *N* in the range `[0,last2-first2)`, or *last1* if no such iterator exists.

Searches the range `[first1,last1)` for a sub-sequence that compares equal value-by-value with the sequence given by `[first2,last2)`, using `predicate` to determine equality, and returns an iterator to the first element of the sub-sequence, or *last1* if no such iterator exists.

See also

`search(_ForwardIter1, _ForwardIter1, _ForwardIter2, _ForwardIter2)`

Definition at line 4615 of file `stl_algo.h`.

2.28.1.24 `template<typename _ForwardIterator , typename _Integer ,
typename _Tp , typename _BinaryPredicate > _ForwardIterator
std::search_n (_ForwardIterator __first, _ForwardIterator
__last, _Integer __count, const _Tp & __val, _BinaryPredicate
__binary_pred)`

Search a sequence for a number of consecutive values using a predicate.

Parameters

first A forward iterator.
last A forward iterator.
count The number of consecutive values.
val The value to find.
binary_pred A binary predicate.

Returns

The first iterator *i* in the range `[first,last-count)` such that `binary_pred(*(i+N),val)` is true for each *N* in the range `[0,count)`, or *last* if no such iterator exists.

Searches the range `[first,last)` for `count` consecutive elements for which the predicate returns true.

Definition at line 4725 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__search_n()`.

2.28.1.25 `template<typename _ForwardIterator , typename _Integer ,
 typename _Tp > _ForwardIterator std::search_n (_ForwardIterator
 __first, _ForwardIterator __last, _Integer __count, const _Tp &
 __val)`

Search a sequence for a number of consecutive values.

Parameters

first A forward iterator.
last A forward iterator.
count The number of consecutive values.
val The value to find.

Returns

The first iterator *i* in the range `[first,last-count)` such that `*(i+N) == val` for each *N* in the range `[0,count)`, or *last* if no such iterator exists.

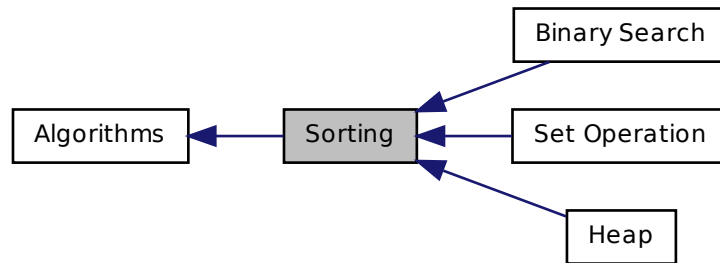
Searches the range `[first,last)` for `count` consecutive elements equal to `val`.

Definition at line 4688 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__search_n()`.

2.29 Sorting

Collaboration diagram for Sorting:



Modules

- [Set Operation](#)
- [Binary Search](#)
- [Heap](#)

Functions

- `template<typename _BidirectionalIterator >`
`void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator >`
`bool std::is_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`bool std::is_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator >`
`_ForwardIterator std::is_sorted_until (_ForwardIterator __first, _ForwardIterator __last)`

- `template<typename _ForwardIterator, typename _Compare >`
`_ForwardIterator std::is_sorted_until (_ForwardIterator __first, _ForwardIterator`
`__last, _Compare __comp)`
- `template<typename _II1, typename _II2 >`
`bool std::lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2`
`__last2)`
- `template<typename _II1, typename _II2, typename _Compare >`
`bool std::lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2`
`__last2, _Compare __comp)`
- `template<typename _Tp >`
`const _Tp & std::max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`
`const _Tp & std::max (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _ForwardIterator >`
`_ForwardIterator std::max_element (_ForwardIterator __first, _ForwardIterator`
`__last)`
- `template<typename _ForwardIterator, typename _Compare >`
`_ForwardIterator std::max_element (_ForwardIterator __first, _ForwardIterator`
`__last, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _`
`InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`
`name _Compare >`
`_OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1,`
`_InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _`
`Compare __comp)`
- `template<typename _Tp, typename _Compare >`
`const _Tp & std::min (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`
`const _Tp & std::min (const _Tp &__a, const _Tp &__b)`
- `template<typename _ForwardIterator >`
`_ForwardIterator std::min_element (_ForwardIterator __first, _ForwardIterator`
`__last)`
- `template<typename _ForwardIterator, typename _Compare >`
`_ForwardIterator std::min_element (_ForwardIterator __first, _ForwardIterator`
`__last, _Compare __comp)`
- `template<typename _Tp, typename _Compare >`
`pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp`
`&__b, _Compare __comp)`
- `template<typename _Tp >`
`pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp`
`&__b)`

- `template<typename _ForwardIterator >`
`pair< _ForwardIterator, _ForwardIterator > std::minmax_element (_-`
`ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`pair< _ForwardIterator, _ForwardIterator > std::minmax_element (_-`
`ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _BidirectionalIterator >`
`bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator`
`__last)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator`
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator`
`__nth, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator`
`__nth, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator`
`__middle, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator`
`__middle, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare >`
`_RandomAccessIterator std::partial_sort_copy (_InputIterator __-`
`first, _InputIterator __last, _RandomAccessIterator __result_first, _-`
`RandomAccessIterator __result_last, _Compare __comp)`
- `template<typename _InputIterator, typename _RandomAccessIterator >`
`_RandomAccessIterator std::partial_sort_copy (_InputIterator __-`
`first, _InputIterator __last, _RandomAccessIterator __result_first, _-`
`RandomAccessIterator __result_last)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`bool std::prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator`
`__last, _Compare __comp)`
- `template<typename _BidirectionalIterator >`
`bool std::prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator`
`__last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last,`
`_Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`

- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::stable_sort (_RandomAccessIterator __first, _RandomAccessIterator`
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::stable_sort (_RandomAccessIterator __first, _RandomAccessIterator`
`__last)`

2.29.1 Function Documentation

2.29.1.1 `template<typename _BidirectionalIterator > void std::inplace_merge`
`(_BidirectionalIterator __first, _BidirectionalIterator __middle,`
`_BidirectionalIterator __last)`

Merges two sorted ranges in place.

Parameters

first An iterator.

middle Another iterator.

last Another iterator.

Returns

Nothing.

Merges two sorted and consecutive ranges, [first,middle) and [middle,last), and puts the result in [first,last). The output will be sorted. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

If enough additional memory is available, this takes (last-first)-1 comparisons. Otherwise an NlogN algorithm is used, where N is distance(first,last).

Definition at line 3058 of file stl_algo.h.

References `std::__merge_adaptive()`, `std::__merge_without_buffer()`, `std::_Temporary_buffer< _ForwardIterator, _Tp >::begin()`, `std::distance()`, and `std::_Temporary_buffer< _ForwardIterator, _Tp >::size()`.

2.29.1.2 `template<typename _BidirectionalIterator, typename _Compare`
`> void std::inplace_merge (_BidirectionalIterator __first,`
`_BidirectionalIterator __middle, _BidirectionalIterator __last,`
`_Compare __comp)`

Merges two sorted ranges in place.

Parameters

first An iterator.
middle Another iterator.
last Another iterator.
comp A functor to use for comparisons.

Returns

Nothing.

Merges two sorted and consecutive ranges, [first,middle) and [middle,last), and puts the result in [first,last). The output will be sorted. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

If enough additional memory is available, this takes (last-first)-1 comparisons. Otherwise an NlogN algorithm is used, where N is distance(first,last).

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 3113 of file stl_algo.h.

References std::__merge_adaptive(), std::__merge_without_buffer(), std::_Temporary_buffer< _ForwardIterator, _Tp >::begin(), std::distance(), and std::_Temporary_buffer< _ForwardIterator, _Tp >::size().

2.29.1.3 `template<typename _ForwardIterator> bool std::is_sorted (_ForwardIterator __first, _ForwardIterator __last) [inline]`

Determines whether the elements of a sequence are sorted.

Parameters

first An iterator.
last Another iterator.

Returns

True if the elements are sorted, false otherwise.

Definition at line 3813 of file stl_algo.h.

References std::is_sorted_until().

2.29.1.4 `template<typename _ForwardIterator, typename _Compare> bool
std::is_sorted (_ForwardIterator __first, _ForwardIterator __last,
_Compare __comp) [inline]`

Determines whether the elements of a sequence are sorted according to a comparison functor.

Parameters

first An iterator.
last Another iterator.
comp A comparison functor.

Returns

True if the elements are sorted, false otherwise.

Definition at line 3827 of file `stl_algo.h`.

References `std::is_sorted_until()`.

2.29.1.5 `template<typename _ForwardIterator> _ForwardIterator
std::is_sorted_until (_ForwardIterator __first, _ForwardIterator
__last)`

Determines the end of a sorted sequence.

Parameters

first An iterator.
last Another iterator.

Returns

An iterator pointing to the last iterator *i* in `[first, last)` for which the range `[first, i)` is sorted.

Definition at line 3841 of file `stl_algo.h`.

2.29.1.6 `template<typename _ForwardIterator, typename _Compare>
_ForwardIterator std::is_sorted_until (_ForwardIterator __first,
_ForwardIterator __last, _Compare __comp)`

Determines the end of a sorted sequence using comparison functor.

Parameters

first An iterator.

last Another iterator.

comp A comparison functor.

Returns

An iterator pointing to the last iterator *i* in [*first*, *last*) for which the range [*first*, *i*) is sorted.

Definition at line 3870 of file `stl_algo.h`.

Referenced by `std::is_sorted()`.

```
2.29.1.7  template<typename _II1 , typename _II2 > bool  
           std::lexicographical_compare ( _II1 __first1, _II1 __last1, _II2  
           __first2, _II2 __last2 ) [inline]
```

Performs **dictionary** comparison on ranges.

Parameters

first1 An input iterator.

last1 An input iterator.

first2 An input iterator.

last2 An input iterator.

Returns

A boolean true or false.

*Returns true if the sequence of elements defined by the range [*first1*,*last1*) is lexicographically less than the sequence of elements defined by the range [*first2*,*last2*). Returns false otherwise.* (Quoted from [25.3.8]/1.) If the iterators are all character pointers, then this is an inline call to `memcmp`.

Definition at line 1073 of file `stl_algobase.h`.

```
2.29.1.8 template<typename _II1 , typename _II2 , typename _Compare >
        bool std::lexicographical_compare ( _II1 __first1, _II1 __last1, _II2
        __first2, _II2 __last2, _Compare __comp )
```

Performs **dictionary** comparison on ranges.

Parameters

first1 An input iterator.
last1 An input iterator.
first2 An input iterator.
last2 An input iterator.
comp A [comparison functor](#).

Returns

A boolean true or false.

The same as the four-parameter `lexicographical_compare`, but uses the `comp` parameter instead of `<`.

Definition at line 1107 of file `stl_algobase.h`.

Referenced by `std::operator<()`.

```
2.29.1.9 template<typename _Tp > const _Tp & std::max ( const _Tp & __a,
        const _Tp & __b ) [inline]
```

This does what you think it does.

Parameters

a A thing of arbitrary type.
b Another thing of arbitrary type.

Returns

The greater of the parameters.

This is the simple classic generic implementation. It will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 210 of file `stl_algobase.h`.

Referenced by `std::_Deque_base<_Tp, _Alloc>::_M_initialize_map()`, `std::deque<_Tp, _Alloc>::_M_reallocate_map()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`.

2.29.1.10 `template<typename _Tp, typename _Compare> const _Tp &
std::max (const _Tp & __a, const _Tp & __b, _Compare __comp
) [inline]`

This does what you think it does.

Parameters

- a* A thing of arbitrary type.
- b* Another thing of arbitrary type.
- comp* A [comparison functor](#).

Returns

The greater of the parameters.

This will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 254 of file `stl_algobase.h`.

2.29.1.11 `template<typename _ForwardIterator> _ForwardIterator
std::max_element (_ForwardIterator __first, _ForwardIterator
__last)`

Return the maximum element in a range.

Parameters

- first* Start of range.
- last* End of range.

Returns

Iterator referencing the first instance of the largest value.

Definition at line 6125 of file `stl_algo.h`.

2.29.1.12 `template<typename _ForwardIterator, typename _Compare >
_ForwardIterator std::max_element (_ForwardIterator __first,
_ForwardIterator __last, _Compare __comp)`

Return the maximum element in a range using comparison functor.

Parameters

first Start of range.
last End of range.
comp Comparison functor.

Returns

Iterator referencing the first instance of the largest value according to comp.

Definition at line 6153 of file `stl_algo.h`.

Referenced by `std::valarray<_Tp>::max()`.

2.29.1.13 `template<typename _InputIterator1, typename _InputIterator2
, typename _OutputIterator > _OutputIterator std::merge (
_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2
__first2, _InputIterator2 __last2, _OutputIterator __result)`

Merges two sorted ranges.

Parameters

first1 An iterator.
first2 Another iterator.
last1 Another iterator.
last2 Another iterator.
result An iterator pointing to the end of the merged range.

Returns

An iterator pointing to the first element *not less than val*.

Merges the ranges `[first1,last1)` and `[first2,last2)` into the sorted range `[result, result + (last1-first1) + (last2-first2))`. Both input ranges must be sorted, and the output range must not overlap with either of the input ranges. The sort is *stable*, that is, for equivalent

elements in the two ranges, elements from the first range will always come before elements from the second.

Definition at line 5368 of file `stl_algo.h`.

```
2.29.1.14  template<typename _InputIterator1 , typename _InputIterator2 ,
            typename _OutputIterator , typename _Compare > _OutputIterator
std::merge ( _InputIterator1 __first1, _InputIterator1 __last1,
            _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator
            __result, _Compare __comp )
```

Merges two sorted ranges.

Parameters

first1 An iterator.

first2 Another iterator.

last1 Another iterator.

last2 Another iterator.

result An iterator pointing to the end of the merged range.

comp A functor to use for comparisons.

Returns

An iterator pointing to the first element "not less than" *val*.

Merges the ranges `[first1,last1)` and `[first2,last2)` into the sorted range `[result, result + (last1-first1) + (last2-first2))`. Both input ranges must be sorted, and the output range must not overlap with either of the input ranges. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 5431 of file `stl_algo.h`.

```
2.29.1.15  template<typename _Tp , typename _Compare > const _Tp &
std::min ( const _Tp & __a, const _Tp & __b, _Compare __comp )
[inline]
```

This does what you think it does.

Parameters

- a* A thing of arbitrary type.
- b* Another thing of arbitrary type.
- comp* A [comparison functor](#).

Returns

The lesser of the parameters.

This will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 233 of file `stl_algobase.h`.

2.29.1.16 `template<typename _Tp> const _Tp & std::min (const _Tp & __a,
const _Tp & __b) [inline]`

This does what you think it does.

Parameters

- a* A thing of arbitrary type.
- b* Another thing of arbitrary type.

Returns

The lesser of the parameters.

This is the simple classic generic implementation. It will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 187 of file `stl_algobase.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, `__gnu_profile::__report()`, `__gnu_parallel::__sequential_random_shuffle()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`, `std::basic_string< _CharT, _Traits, _Alloc >::compare()`, `std::basic_string< char >::compare()`, `std::generate_canonical()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `__gnu_cxx::random_sample_n()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rfind()`, `std::basic_string< _CharT, _Traits, _Alloc >::rfind()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, `std::basic_streambuf< _CharT, _Traits >::xsgetn()`, `std::basic_streambuf< _CharT, _Traits >::xsputn()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

2.29.1.17 `template<typename _ForwardIterator > _ForwardIterator
std::min_element (_ForwardIterator __first, _ForwardIterator
__last)`

Return the minimum element in a range.

Parameters

first Start of range.

last End of range.

Returns

Iterator referencing the first instance of the smallest value.

Definition at line 6069 of file stl_algo.h.

2.29.1.18 `template<typename _ForwardIterator , typename _Compare >
_ForwardIterator std::min_element (_ForwardIterator __first,
_ForwardIterator __last, _Compare __comp)`

Return the minimum element in a range using comparison functor.

Parameters

first Start of range.

last End of range.

comp Comparison functor.

Returns

Iterator referencing the first instance of the smallest value according to comp.

Definition at line 6097 of file stl_algo.h.

Referenced by std::valarray< _Tp >::min().

2.29.1.19 `template<typename _Tp , typename _Compare > pair< const _Tp
&, const _Tp & > std::minmax (const _Tp & __a, const _Tp &
__b, _Compare __comp) [inline]`

Determines min and max at once as an ordered pair.

Parameters

- a* A thing of arbitrary type.
- b* Another thing of arbitrary type.
- comp* A [comparison functor](#).

Returns

A pair(b, a) if b is smaller than a, pair(a, b) otherwise.

Definition at line 3918 of file `stl_algo.h`.

2.29.1.20 `template<typename _Tp > pair< const _Tp &, const _Tp & >
std::minmax (const _Tp & __a, const _Tp & __b) [inline]`

Determines min and max at once as an ordered pair.

Parameters

- a* A thing of arbitrary type.
- b* Another thing of arbitrary type.

Returns

A pair(b, a) if b is smaller than a, pair(a, b) otherwise.

Definition at line 3899 of file `stl_algo.h`.

2.29.1.21 `template<typename _ForwardIterator > pair<_ForwardIterator,
_ForwardIterator> std::minmax_element (_ForwardIterator
__first, _ForwardIterator __last)`

Return a pair of iterators pointing to the minimum and maximum elements in a range.

Parameters

- first* Start of range.
- last* End of range.

Returns

make_pair(m, M), where m is the first iterator i in [first, last) such that no other element in the range is smaller, and where M is the last iterator i in [first, last) such that no other element in the range is larger.

Definition at line 3937 of file stl_algo.h.

References std::make_pair().

2.29.1.22 `template<typename _ForwardIterator, typename _Compare >
pair<_ForwardIterator, _ForwardIterator> std::minmax_element (
_ForwardIterator __first, _ForwardIterator __last, _Compare
__comp)`

Return a pair of iterators pointing to the minimum and maximum elements in a range.

Parameters

first Start of range.

last End of range.

comp Comparison functor.

Returns

make_pair(m, M), where m is the first iterator i in [first, last) such that no other element in the range is smaller, and where M is the last iterator i in [first, last) such that no other element in the range is larger.

Definition at line 4013 of file stl_algo.h.

References std::make_pair().

2.29.1.23 `template<typename _BidirectionalIterator > bool
std::next_permutation (_BidirectionalIterator __first,
_BidirectionalIterator __last)`

Permute range into the next *dictionary* ordering.

Parameters

first Start of range.

last End of range.

Returns

False if wrapped to first permutation, true otherwise.

Treats all permutations of the range as a set of *dictionary* sorted sequences. Permutes the current sequence into the next one of this set. Returns true if there are more sequences to generate. If the sequence is the largest of the set, the smallest is generated and false returned.

Definition at line 3515 of file stl_algo.h.

References std::iter_swap(), and std::reverse().

2.29.1.24 `template<typename _BidirectionalIterator, typename _Compare>
> bool std::next_permutation (_BidirectionalIterator __first,
_BidirectionalIterator __last, _Compare __comp)`

Permute range into the next *dictionary* ordering using comparison functor.

Parameters

first Start of range.

last End of range.

comp A comparison functor.

Returns

False if wrapped to first permutation, true otherwise.

Treats all permutations of the range [first,last) as a set of *dictionary* sorted sequences ordered by *comp*. Permutes the current sequence into the next one of this set. Returns true if there are more sequences to generate. If the sequence is the largest of the set, the smallest is generated and false returned.

Definition at line 3572 of file stl_algo.h.

References std::iter_swap(), and std::reverse().

2.29.1.25 `template<typename _RandomAccessIterator> void std::nth_element
(_RandomAccessIterator __first, _RandomAccessIterator __nth,
_RandomAccessIterator __last) [inline]`

Sort a sequence just enough to find a particular position.

Parameters

first An iterator.

nth Another iterator.

last Another iterator.

Returns

Nothing.

Rearranges the elements in the range `[first,last)` so that `*nth` is the same element that would have been in that position had the whole sequence been sorted. whole sequence been sorted. The elements either side of `*nth` are not completely sorted, but for any iterator in the range `[first,nth)` and any iterator in the range `[nth,last)` it holds that `*j < *i` is false.

Definition at line 5213 of file `stl_algo.h`.

References `std::__lg()`.

```
2.29.1.26  template<typename _RandomAccessIterator , typename _Compare
> void std::nth_element ( _RandomAccessIterator __first,
    _RandomAccessIterator __nth, _RandomAccessIterator __last,
    _Compare __comp ) [inline]
```

Sort a sequence just enough to find a particular position using a predicate for comparison.

Parameters

first An iterator.

nth Another iterator.

last Another iterator.

comp A comparison functor.

Returns

Nothing.

Rearranges the elements in the range `[first,last)` so that `*nth` is the same element that would have been in that position had the whole sequence been sorted. The elements either side of `*nth` are not completely sorted, but for any iterator in the range `[first,nth)` and any iterator in the range `[nth,last)` it holds that `comp(*j, *i)` is false.

Definition at line 5252 of file `stl_algo.h`.

References `std::__lg()`.

2.29.1.27 `template<typename _RandomAccessIterator > void std::partial_sort
(_RandomAccessIterator __first, _RandomAccessIterator
__middle, _RandomAccessIterator __last) [inline]`

Sort the smallest elements of a sequence.

Parameters

first An iterator.

middle Another iterator.

last Another iterator.

Returns

Nothing.

Sorts the smallest (middle-first) elements in the range [first,last) and moves them to the range [first,middle). The order of the remaining elements in the range [middle,last) is undefined. After the sort if *i* and *j* are iterators in the range [first,middle) such that *j* precedes *i* and *k* is an iterator in the range [middle,last) then **j* < **i* and **k* < **i* are both false.

Definition at line 5136 of file stl_algo.h.

References std::__heap_select(), and std::sort_heap().

2.29.1.28 `template<typename _RandomAccessIterator , typename _Compare
> void std::partial_sort (_RandomAccessIterator __first,
_RandomAccessIterator __middle, _RandomAccessIterator __last,
_Compare __comp) [inline]`

Sort the smallest elements of a sequence using a predicate for comparison.

Parameters

first An iterator.

middle Another iterator.

last Another iterator.

comp A comparison functor.

Returns

Nothing.

Sorts the smallest (middle-first) elements in the range `[first,last)` and moves them to the range `[first,middle)`. The order of the remaining elements in the range `[middle,last)` is undefined. After the sort if `i` and `j` are iterators in the range `[first,middle)` such that `i` precedes `j` and `i` is an iterator in the range `[middle,last)` then `*comp(j,*i)` and `comp(*j,*i)` are both false.

Definition at line 5175 of file `stl_algo.h`.

References `std::__heap_select()`, and `std::sort_heap()`.

2.29.1.29 `template<typename _InputIterator , typename
_RandomAccessIterator , typename _Compare >
_RandomAccessIterator std::partial_sort_copy (_InputIterator
__first, _InputIterator __last, _RandomAccessIterator __result_first,
_RandomAccessIterator __result_last, _Compare __comp)`

Copy the smallest elements of a sequence using a predicate for comparison.

Parameters

first An input iterator.

last Another input iterator.

result_first A random-access iterator.

result_last Another random-access iterator.

comp A comparison functor.

Returns

An iterator indicating the end of the resulting sequence.

Copies and sorts the smallest `N` values from the range `[first,last)` to the range beginning at `result_first`, where the number of elements to be copied, `N`, is the smaller of `(last-first)` and `(result_last-result_first)`. After the sort if `i` and `j` are iterators in the range `[result_first,result_first+N)` such that `i` precedes `j` then `comp(*j,*i)` is false. The value returned is `result_first+N`.

Definition at line 2010 of file `stl_algo.h`.

References `std::make_heap()`, and `std::sort_heap()`.

2.29.1.30 `template<typename _InputIterator , typename
_RandomAccessIterator > _RandomAccessIterator
std::partial_sort_copy (_InputIterator __first,
_InputIterator __last, _RandomAccessIterator __result_first,
_RandomAccessIterator __result_last)`

Copy the smallest elements of a sequence.

Parameters

first An iterator.

last Another iterator.

result_first A random-access iterator.

result_last Another random-access iterator.

Returns

An iterator indicating the end of the resulting sequence.

Copies and sorts the smallest N values from the range [first,last) to the range beginning at result_first, where the number of elements to be copied, N, is the smaller of (last-first) and (result_last-result_first). After the sort if i and are iterators in the range [result_first,result_first+N) such that precedes then *j<*i is false. The value returned is result_first+N.

Definition at line 1944 of file stl_algo.h.

References std::make_heap(), and std::sort_heap().

2.29.1.31 `template<typename _BidirectionalIterator , typename _Compare
> bool std::prev_permutation (_BidirectionalIterator __first,
_BidirectionalIterator __last, _Compare __comp)`

Permute range into the previous *dictionary* ordering using comparison functor.

Parameters

first Start of range.

last End of range.

comp A comparison functor.

Returns

False if wrapped to last permutation, true otherwise.

Treats all permutations of the range `[first,last)` as a set of *dictionary* sorted sequences ordered by *comp*. Permutes the current sequence into the previous one of this set. Returns true if there are more sequences to generate. If the sequence is the smallest of the set, the largest is generated and false returned.

Definition at line 3685 of file `stl_algo.h`.

References `std::iter_swap()`, and `std::reverse()`.

2.29.1.32 `template<typename _BidirectionalIterator > bool
std::prev_permutation (_BidirectionalIterator __first,
_BidirectionalIterator __last)`

Permute range into the previous *dictionary* ordering.

Parameters

first Start of range.

last End of range.

Returns

False if wrapped to last permutation, true otherwise.

Treats all permutations of the range as a set of *dictionary* sorted sequences. Permutes the current sequence into the previous one of this set. Returns true if there are more sequences to generate. If the sequence is the smallest of the set, the largest is generated and false returned.

Definition at line 3628 of file `stl_algo.h`.

References `std::iter_swap()`, and `std::reverse()`.

2.29.1.33 `template<typename _RandomAccessIterator , typename
_Compare > void std::sort (_RandomAccessIterator __first,
_RandomAccessIterator __last, _Compare __comp) [inline]`

Sort the elements of a sequence using a predicate for comparison.

Parameters

first An iterator.

last Another iterator.

comp A comparison functor.

Returns

Nothing.

Sorts the elements in the range `[first,last)` in ascending order, such that `comp(*(i+1),*i)` is false for every iterator `i` in the range `[first,last-1)`.

The relative ordering of equivalent elements is not preserved, use `stable_sort()` if this is needed.

Definition at line 5326 of file `stl_algo.h`.

References `std::__final_insertion_sort()`, `std::__introsort_loop()`, and `std::__lg()`.

2.29.1.34 `template<typename _RandomAccessIterator > void std::sort (`
`_RandomAccessIterator __first, _RandomAccessIterator __last)`
`[inline]`

Sort the elements of a sequence.

Parameters

first An iterator.

last Another iterator.

Returns

Nothing.

Sorts the elements in the range `[first,last)` in ascending order, such that `*(i+1) < *i` is false for each iterator `i` in the range `[first,last-1)`.

The relative ordering of equivalent elements is not preserved, use `stable_sort()` if this is needed.

Definition at line 5290 of file `stl_algo.h`.

References `std::__final_insertion_sort()`, `std::__introsort_loop()`, and `std::__lg()`.

2.29.1.35 `template<typename _RandomAccessIterator , typename _Compare`
`> void std::stable_sort (_RandomAccessIterator __first,`
`_RandomAccessIterator __last, _Compare __comp) [inline]`

Sort the elements of a sequence using a predicate for comparison, preserving the relative order of equivalent elements.

Parameters

first An iterator.
last Another iterator.
comp A comparison functor.

Returns

Nothing.

Sorts the elements in the range `[first,last)` in ascending order, such that `comp(*(i+1),*i)` is false for each iterator `i` in the range `[first,last-1)`.

The relative ordering of equivalent elements is preserved, so any two elements `x` and `y` in the range `[first,last)` such that `comp(x,y)` is false and `comp(y,x)` is false will have the same relative ordering after calling `stable_sort()`.

Definition at line 5532 of file `stl_algo.h`.

References `std::__inplace_stable_sort()`, `std::_Temporary_buffer<_ForwardIterator, _Tp>::begin()`, and `std::_Temporary_buffer<_ForwardIterator, _Tp>::size()`.

2.29.1.36 `template<typename _RandomAccessIterator> void std::stable_sort
 (_RandomAccessIterator __first, _RandomAccessIterator __last)
 [inline]`

Sort the elements of a sequence, preserving the relative order of equivalent elements.

Parameters

first An iterator.
last Another iterator.

Returns

Nothing.

Sorts the elements in the range `[first,last)` in ascending order, such that `*(i+1)<*i` is false for each iterator `i` in the range `[first,last-1)`.

The relative ordering of equivalent elements is preserved, so any two elements `x` and `y` in the range `[first,last)` such that `x<y` is false and `y<x` is false will have the same relative ordering after calling `stable_sort()`.

Definition at line 5490 of file `stl_algo.h`.

References `std::__inplace_stable_sort()`, `std::_Temporary_buffer<_ForwardIterator, _Tp>::begin()`, and `std::_Temporary_buffer<_ForwardIterator, _Tp>::size()`.

2.30 Set Operation

Collaboration diagram for Set Operation:



Functions

- `template<typename _InputIterator1, typename _InputIterator2 >`
`bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _-`
`_InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare >`
`bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _-`
`_InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`
`name _Compare >`
`_OutputIterator std::set_difference (_InputIterator1 __first1, _InputIterator1 __-`
`last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result,`
`_Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator std::set_difference (_InputIterator1 __first1, _InputIterator1 __-`
`last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator std::set_intersection (_InputIterator1 __first1, _InputIterator1`
`__last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __-`
`result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`
`name _Compare >`
`_OutputIterator std::set_intersection (_InputIterator1 __first1, _InputIterator1`
`__last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __-`
`result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator std::set_symmetric_difference (_InputIterator1 __first1, _-`
`_InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _-`
`_OutputIterator __result)`

- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator std::set_symmetric_difference (_InputIterator1 __first1, _-`
`InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _-`
`OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator std::set_union (_InputIterator1 __first1, _InputIterator1 __last1,`
`_InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`
`name _Compare >`
`_OutputIterator std::set_union (_InputIterator1 __first1, _InputIterator1 __last1,`
`_InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _-`
`Compare __comp)`

2.30.1 Detailed Description

These algorithms are common set operations performed on sequences that are already sorted. The number of comparisons will be linear.

2.30.2 Function Documentation

2.30.2.1 `template<typename _InputIterator1, typename _InputIterator2 >`
`bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1,`
`_InputIterator2 __first2, _InputIterator2 __last2)`

Determines whether all elements of a sequence exists in a range.

Parameters

first1 Start of search range.

last1 End of search range.

first2 Start of sequence

last2 End of sequence.

Returns

True if each element in [first2,last2) is contained in order within [first1,last1). False otherwise.

This operation expects both [first1,last1) and [first2,last2) to be sorted. Searches for the presence of each element in [first2,last2) within [first1,last1). The iterators over each range only move forward, so this is a linear algorithm. If an element in [first2,last2) is not found before the search iterator reaches *last2*, false is returned.

Definition at line 3411 of file stl_algo.h.

```
2.30.2.2  template<typename _InputIterator1 , typename _InputIterator2 ,
            typename _Compare > bool std::includes ( _InputIterator1 __first1,
            _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2
            __last2, _Compare __comp )
```

Determines whether all elements of a sequence exists in a range using comparison.

Parameters

first1 Start of search range.
last1 End of search range.
first2 Start of sequence
last2 End of sequence.
comp Comparison function to use.

Returns

True if each element in [first2,last2) is contained in order within [first1,last1) according to comp. False otherwise.

This operation expects both [first1,last1) and [first2,last2) to be sorted. Searches for the presence of each element in [first2,last2) within [first1,last1), using comp to decide. The iterators over each range only move forward, so this is a linear algorithm. If an element in [first2,last2) is not found before the search iterator reaches *last2*, false is returned.

Definition at line 3461 of file stl_algo.h.

```
2.30.2.3  template<typename _InputIterator1 , typename _InputIterator2 ,
            typename _OutputIterator , typename _Compare > _OutputIterator
            std::set_difference ( _InputIterator1 __first1, _InputIterator1
            __last1, _InputIterator2 __first2, _InputIterator2 __last2,
            _OutputIterator __result, _Compare __comp )
```

Return the difference of two sorted ranges using comparison functor.

Parameters

first1 Start of first range.

last1 End of first range.
first2 Start of second range.
last2 End of second range.
comp The comparison functor.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in the first range but not the second in order to the output range. Iterators increment for each range. When the current element of the first range is less than the second according to *comp*, that element is copied and the iterator advances. If the current element of the second range is less, no element is copied and the iterator advances. If an element is contained in both ranges according to *comp*, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5889 of file `stl_algo.h`.

2.30.2.4 `template<typename _InputIterator1, typename _InputIterator2 ,
typename _OutputIterator > _OutputIterator std::set_difference (
_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2
__first2, _InputIterator2 __last2, _OutputIterator __result)`

Return the difference of two sorted ranges.

Parameters

first1 Start of first range.
last1 End of first range.
first2 Start of second range.
last2 End of second range.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in the first range but not the second in order to the output range. Iterators increment for each range. When the current element of the first range is less than the second, that element is copied and the iterator advances. If the current element of the second range is less, the iterator advances, but no element is copied. If an element is contained in both ranges, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5828 of file `stl_algo.h`.

2.30.2.5 `template<typename _InputIterator1, typename _InputIterator2 ,
typename _OutputIterator > _OutputIterator std::set_intersection (`
 `_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2`
 `__first2, _InputIterator2 __last2, _OutputIterator __result)`

Return the intersection of two sorted ranges.

Parameters

first1 Start of first range.

last1 End of first range.

first2 Start of second range.

last2 End of second range.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in both ranges in order to the output range. Iterators increment for each range. When the current element of one range is less than the other, that iterator advances. If an element is contained in both ranges, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5713 of file `stl_algo.h`.

2.30.2.6 `template<typename _InputIterator1, typename _InputIterator2 ,`
 `typename _OutputIterator, typename _Compare > _OutputIterator`
 `std::set_intersection (_InputIterator1 __first1, _InputIterator1`
 `__last1, _InputIterator2 __first2, _InputIterator2 __last2,`
 `_OutputIterator __result, _Compare __comp)`

Return the intersection of two sorted ranges using comparison functor.

Parameters

first1 Start of first range.

last1 End of first range.

first2 Start of second range.

last2 End of second range.

comp The comparison functor.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in both ranges in order to the output range. Iterators increment for each range. When the current element of one range is less than the other according to *comp*, that iterator advances. If an element is contained in both ranges according to *comp*, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5770 of file `stl_algo.h`.

```
2.30.2.7  template<typename _InputIterator1 , typename  
           _InputIterator2 , typename _OutputIterator > _OutputIterator  
           std::set_symmetric_difference ( _InputIterator1 __first1,  
           _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2  
           __last2, _OutputIterator __result )
```

Return the symmetric difference of two sorted ranges.

Parameters

first1 Start of first range.

last1 End of first range.

first2 Start of second range.

last2 End of second range.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in one range but not the other in order to the output range. Iterators increment for each range. When the current element of one range is less than the other, that element is copied and the iterator advances. If an element is contained in both ranges, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5947 of file `stl_algo.h`.

2.30.2.8 `template<typename _InputIterator1, typename _InputIterator2 ,
 typename _OutputIterator, typename _Compare > _OutputIterator
 std::set_symmetric_difference (_InputIterator1 __first1,
 _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2
 __last2, _OutputIterator __result, _Compare __comp)`

Return the symmetric difference of two sorted ranges using comparison functor.

Parameters

first1 Start of first range.
last1 End of first range.
first2 Start of second range.
last2 End of second range.
comp The comparison functor.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in one range but not the other in order to the output range. Iterators increment for each range. When the current element of one range is less than the other according to *comp*, that element is copied and the iterator advances. If an element is contained in both ranges according to *comp*, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 6013 of file `stl_algo.h`.

2.30.2.9 `template<typename _InputIterator1, typename _InputIterator2 ,
 typename _OutputIterator > _OutputIterator std::set_union (`
`_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2`
`__first2, _InputIterator2 __last2, _OutputIterator __result)`

Return the union of two sorted ranges.

Parameters

first1 Start of first range.
last1 End of first range.
first2 Start of second range.

last2 End of second range.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in each range in order to the output range. Iterators increment for each range. When the current element of one range is less than the other, that element is copied and the iterator advanced. If an element is contained in both ranges, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5579 of file `stl_algo.h`.

```
2.30.2.10  template<typename _InputIterator1 , typename _InputIterator2 ,
              typename _OutputIterator , typename _Compare > _OutputIterator
              std::set_union ( _InputIterator1 __first1, _InputIterator1 __last1,
                              _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator
                              __result, _Compare __comp )
```

Return the union of two sorted ranges using a comparison functor.

Parameters

first1 Start of first range.

last1 End of first range.

first2 Start of second range.

last2 End of second range.

comp The comparison functor.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in each range in order to the output range. Iterators increment for each range. When the current element of one range is less than the other according to *comp*, that element is copied and the iterator advanced. If an equivalent element according to *comp* is contained in both ranges, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5646 of file `stl_algo.h`.

2.31 Binary Search

Collaboration diagram for Binary Search:



Functions

- `template<typename _ForwardIterator, typename _Tp >`
`bool std::binary_search (_ForwardIterator __first, _ForwardIterator __last, const`
`_Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`bool std::binary_search (_ForwardIterator __first, _ForwardIterator __last, const`
`_Tp &__val, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`pair< _ForwardIterator, _ForwardIterator > std::equal_range (_ForwardIterator`
`__first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp >`
`pair< _ForwardIterator, _ForwardIterator > std::equal_range (_ForwardIterator`
`__first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`_ForwardIterator std::lower_bound (_ForwardIterator __first, _ForwardIterator`
`__last, const _Tp &__val, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp >`
`_ForwardIterator std::lower_bound (_ForwardIterator __first, _ForwardIterator`
`__last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`_ForwardIterator std::upper_bound (_ForwardIterator __first, _ForwardIterator`
`__last, const _Tp &__val, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp >`
`_ForwardIterator std::upper_bound (_ForwardIterator __first, _ForwardIterator`
`__last, const _Tp &__val)`

2.31.1 Detailed Description

These algorithms are variations of a classic binary search, and all assume that the sequence being searched is already sorted.

The number of comparisons will be logarithmic (and as few as possible). The number of steps through the sequence will be logarithmic for random-access iterators (e.g., pointers), and linear otherwise.

The LWG has passed Defect Report 270, which notes: *The proposed resolution reinterprets binary search. Instead of thinking about searching for a value in a sorted range, we view that as an important special case of a more general algorithm: searching for the partition point in a partitioned range. We also add a guarantee that the old wording did not: we ensure that the upper bound is no earlier than the lower bound, that the pair returned by equal_range is a valid range, and that the first part of that pair is the lower bound.*

The actual effect of the first sentence is that a comparison functor passed by the user doesn't necessarily need to induce a strict weak ordering relation. Rather, it partitions the range.

2.31.2 Function Documentation

2.31.2.1 `template<typename _ForwardIterator, typename _Tp> bool
std::binary_search (_ForwardIterator __first, _ForwardIterator
__last, const _Tp & __val)`

Determines whether an element exists in a range.

Parameters

first An iterator.

last Another iterator.

val The search term.

Returns

True if *val* (or its equivalent) is in [*first*,*last*].

Note that this does not actually return an iterator to *val*. For that, use `std::find` or a container's specialized find member functions.

Definition at line 2665 of file `stl_algo.h`.

References `std::lower_bound()`.

2.31.2.2 `template<typename _ForwardIterator , typename _Tp , typename
_Compare > bool std::binary_search (_ForwardIterator __first,
_ForwardIterator __last, const _Tp & __val, _Compare __comp)`

Determines whether an element exists in a range.

Parameters

first An iterator.

last Another iterator.

val The search term.

comp A functor to use for comparisons.

Returns

True if *val* (or its equivalent) is in [*first*,*last*].

Note that this does not actually return an iterator to *val*. For that, use `std::find` or a container's specialized find member functions.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 2698 of file `stl_algo.h`.

References `std::lower_bound()`.

2.31.2.3 `template<typename _ForwardIterator , typename _Tp , typename
_Compare > pair<_ForwardIterator, _ForwardIterator>
std::equal_range (_ForwardIterator __first, _ForwardIterator
__last, const _Tp & __val, _Compare __comp)`

Finds the largest subrange in which *val* could be inserted at any place in it without changing the ordering.

Parameters

first An iterator.

last Another iterator.

val The search term.

comp A functor to use for comparisons.

Returns

An pair of iterators defining the subrange.

This is equivalent to

```
std::make_pair(lower_bound(first, last, val, comp),
               upper_bound(first, last, val, comp))
```

but does not actually call those functions.

Definition at line 2605 of file `stl_algo.h`.

References `std::advance()`, `std::distance()`, `std::lower_bound()`, and `std::upper_bound()`.

2.31.2.4 `template<typename _ForwardIterator, typename _Tp >`
`pair<_ForwardIterator, _ForwardIterator> std::equal_range (`
 `_ForwardIterator __first, _ForwardIterator __last, const _Tp &`
 `__val)`

Finds the largest subrange in which *val* could be inserted at any place in it without changing the ordering.

Parameters

first An iterator.

last Another iterator.

val The search term.

Returns

An pair of iterators defining the subrange.

This is equivalent to

```
std::make_pair(lower_bound(first, last, val),
               upper_bound(first, last, val))
```

but does not actually call those functions.

Definition at line 2543 of file `stl_algo.h`.

References `std::advance()`, `std::distance()`, `std::lower_bound()`, and `std::upper_bound()`.

2.31.2.5 `template<typename _ForwardIterator , typename _Tp , typename
_Compare > _ForwardIterator std::lower_bound (_ForwardIterator
__first, _ForwardIterator __last, const _Tp & __val, _Compare
__comp)`

Finds the first position in which *val* could be inserted without changing the ordering.

Parameters

first An iterator.

last Another iterator.

val The search term.

comp A functor to use for comparisons.

Returns

An iterator pointing to the first element *not less than val*, or `end()` if every element is less than *val*.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 2394 of file `stl_algo.h`.

References `std::advance()`, and `std::distance()`.

2.31.2.6 `template<typename _ForwardIterator , typename _Tp >
_ForwardIterator std::lower_bound (_ForwardIterator __first,
_ForwardIterator __last, const _Tp & __val)`

Finds the first position in which *val* could be inserted without changing the ordering.

Parameters

first An iterator.

last Another iterator.

val The search term.

Returns

An iterator pointing to the first element *not less than val*, or `end()` if every element is less than *val*.

Definition at line 936 of file `stl_algobase.h`.

References `std::advance()`, and `std::distance()`.

Referenced by `std::__merge_adaptive()`, `std::__merge_without_buffer()`, `std::binary_search()`, and `std::equal_range()`.

2.31.2.7 `template<typename _ForwardIterator , typename _Tp , typename
_Compare > _ForwardIterator std::upper_bound (_ForwardIterator
__first, _ForwardIterator __last, const _Tp & __val, _Compare
__comp)`

Finds the last position in which *val* could be inserted without changing the ordering.

Parameters

first An iterator.

last Another iterator.

val The search term.

comp A functor to use for comparisons.

Returns

An iterator pointing to the first element greater than *val*, or `end()` if no elements are greater than *val*.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 2490 of file `stl_algo.h`.

References `std::advance()`, and `std::distance()`.

Referenced by `std::__merge_adaptive()`, `std::__merge_without_buffer()`, and `std::equal_range()`.

2.31.2.8 `template<typename _ForwardIterator , typename _Tp >
_ForwardIterator std::upper_bound (_ForwardIterator __first,
_ForwardIterator __last, const _Tp & __val)`

Finds the last position in which *val* could be inserted without changing the ordering.

Parameters

first An iterator.

last Another iterator.

val The search term.

Returns

An iterator pointing to the first element greater than *val*, or `end()` if no elements are greater than *val*.

Definition at line 2441 of file `stl_algo.h`.

References `std::advance()`, and `std::distance()`.

2.32 Allocators

Collaboration diagram for Allocators:



Classes

- class `__gnu_cxx::__mt_alloc< _Tp, _Poolp >`

This is a fixed size (power of 2) allocator which - when compiled with thread support - will maintain one freelist per size per thread plus a global one. Steps are taken to limit the per thread freelist sizes (by returning excess back to the global list).

Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt12ch32.html>.

- class `__gnu_cxx::__pool_alloc< _Tp >`

Allocator using a memory pool with a single lock.

- class `__gnu_cxx::_ExtPtr_allocator< _Tp >`

An example allocator which uses a non-standard pointer type.

This allocator specifies that containers use a 'relative pointer' as it's pointer type. (See [ext/pointer.h](#)) Memory allocation in this example is still performed using `std::allocator`.

- class `__gnu_cxx::array_allocator< _Tp, _Array >`

An allocator that uses previously allocated memory. This memory can be externally, globally, or otherwise allocated.

- class `__gnu_cxx::bitmap_allocator< _Tp >`
Bitmap Allocator, primary template.
- class `__gnu_cxx::debug_allocator< _Alloc >`
*A meta-allocator with debugging bits, as per [20.4].
This is precisely the allocator defined in the C++ Standard.*
 - all allocation calls operator new
 - all deallocation calls operator delete.
- class `__gnu_cxx::malloc_allocator< _Tp >`
*An allocator that uses malloc.
This is precisely the allocator defined in the C++ Standard.*
 - all allocation calls malloc
 - all deallocation calls free.
- class `__gnu_cxx::new_allocator< _Tp >`
*An allocator that uses global new, as per [20.4].
This is precisely the allocator defined in the C++ Standard.*
 - all allocation calls operator new
 - all deallocation calls operator delete.
- class `__gnu_cxx::throw_allocator_base< _Tp, _Cond >`
*Allocator class with logging and exception generation control. Intended to be used as an allocator_type in templated code.
Note: Deallocate not allowed to throw.*
- class `std::allocator< _Tp >`
*The standard allocator, as per [20.4].
Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt04ch11.html>.*

2.32.1 Detailed Description

Classes encapsulating memory operations.

2.33 Atomics

Classes

- struct `std::__atomic_flag_base`

Base type for atomic_flag.

- struct `std::atomic< _Tp >`
atomic /// 29.4.3, Generic atomic type, primary class template.
- struct `std::atomic< _Tp * >`
Partial specialization for pointer types.
- struct `std::atomic< bool >`
Explicit specialization for bool.
- struct `std::atomic< char >`
Explicit specialization for char.
- struct `std::atomic< char16_t >`
Explicit specialization for char16_t.
- struct `std::atomic< char32_t >`
Explicit specialization for char32_t.
- struct `std::atomic< int >`
Explicit specialization for int.
- struct `std::atomic< long >`
Explicit specialization for long.
- struct `std::atomic< long long >`
Explicit specialization for long long.
- struct `std::atomic< short >`
Explicit specialization for short.
- struct `std::atomic< signed char >`
Explicit specialization for signed char.
- struct `std::atomic< unsigned char >`
Explicit specialization for unsigned char.
- struct `std::atomic< unsigned int >`
Explicit specialization for unsigned int.
- struct `std::atomic< unsigned long >`

Explicit specialization for unsigned long.

- struct `std::atomic< unsigned long long >`
Explicit specialization for unsigned long long.
- struct `std::atomic< unsigned short >`
Explicit specialization for unsigned short.
- struct `std::atomic< wchar_t >`
Explicit specialization for wchar_t.
- struct `std::atomic_bool`
atomic_bool

Defines

- #define `_GLIBCXX_ATOMIC_NAMESPACE`
- #define `_GLIBCXX_ATOMIC_PROPERTY`
- #define `ATOMIC_ADDRESS_LOCK_FREE`
- #define `ATOMIC_CHAR16_T_LOCK_FREE`
- #define `ATOMIC_CHAR32_T_LOCK_FREE`
- #define `ATOMIC_CHAR_LOCK_FREE`
- #define `ATOMIC_FLAG_INIT`
- #define `ATOMIC_INT_LOCK_FREE`
- #define `ATOMIC_LLONG_LOCK_FREE`
- #define `ATOMIC_LONG_LOCK_FREE`
- #define `ATOMIC_SHORT_LOCK_FREE`
- #define `ATOMIC_VAR_INIT(_VI)`
- #define `ATOMIC_WCHAR_T_LOCK_FREE`

Typedefs

- typedef `__atomic_base< char >` `std::atomic_char`
- typedef `__atomic_base< char16_t >` `std::atomic_char16_t`
- typedef `__atomic_base< char32_t >` `std::atomic_char32_t`
- typedef `__atomic_base< int >` `std::atomic_int`
- typedef `__atomic_base< int_fast16_t >` `std::atomic_int_fast16_t`
- typedef `__atomic_base< int_fast32_t >` `std::atomic_int_fast32_t`
- typedef `__atomic_base< int_fast64_t >` `std::atomic_int_fast64_t`
- typedef `__atomic_base< int_fast8_t >` `std::atomic_int_fast8_t`
- typedef `__atomic_base< int_least16_t >` `std::atomic_int_least16_t`

- typedef __atomic_base< int_least32_t > [std::atomic_int_least32_t](#)
- typedef __atomic_base< int_least64_t > [std::atomic_int_least64_t](#)
- typedef __atomic_base< int_least8_t > [std::atomic_int_least8_t](#)
- typedef __atomic_base< intmax_t > [std::atomic_intmax_t](#)
- typedef __atomic_base< intptr_t > [std::atomic_intptr_t](#)
- typedef __atomic_base< long long > [std::atomic_llong](#)
- typedef __atomic_base< long > [std::atomic_long](#)
- typedef __atomic_base< ptrdiff_t > [std::atomic_ptrdiff_t](#)
- typedef __atomic_base< signed char > [std::atomic_schar](#)
- typedef __atomic_base< short > [std::atomic_short](#)
- typedef __atomic_base< size_t > [std::atomic_size_t](#)
- typedef __atomic_base< unsigned char > [std::atomic_uchar](#)
- typedef __atomic_base< unsigned int > [std::atomic_uint](#)
- typedef __atomic_base< uint_fast16_t > [std::atomic_uint_fast16_t](#)
- typedef __atomic_base< uint_fast32_t > [std::atomic_uint_fast32_t](#)
- typedef __atomic_base< uint_fast64_t > [std::atomic_uint_fast64_t](#)
- typedef __atomic_base< uint_fast8_t > [std::atomic_uint_fast8_t](#)
- typedef __atomic_base< uint_least16_t > [std::atomic_uint_least16_t](#)
- typedef __atomic_base< uint_least32_t > [std::atomic_uint_least32_t](#)
- typedef __atomic_base< uint_least64_t > [std::atomic_uint_least64_t](#)
- typedef __atomic_base< uint_least8_t > [std::atomic_uint_least8_t](#)
- typedef __atomic_base< uintmax_t > [std::atomic_uintmax_t](#)
- typedef __atomic_base< uintptr_t > [std::atomic_uintptr_t](#)
- typedef __atomic_base< unsigned long long > [std::atomic_ullong](#)
- typedef __atomic_base< unsigned long > [std::atomic_ulong](#)
- typedef __atomic_base< unsigned short > [std::atomic_ushort](#)
- typedef __atomic_base< wchar_t > [std::atomic_wchar_t](#)
- typedef enum [std::memory_order](#) [std::memory_order](#)

Enumerations

- enum [std::memory_order](#) {
memory_order_relaxed, memory_order_consume, memory_order_
acquire, memory_order_release,
memory_order_acq_rel, memory_order_seq_cst }

Functions

- `memory_order std::__calculate_memory_order (memory_order __m)`
- `bool std::atomic_compare_exchange_strong (volatile atomic_address *__a, void **__v1, void *__v2)`
- `bool std::atomic_compare_exchange_strong (volatile atomic_bool *__a, bool *__i1, bool __i2)`
- `bool std::atomic_compare_exchange_strong (atomic_address *__a, void **__v1, void *__v2)`
- `bool std::atomic_compare_exchange_strong (atomic_bool *__a, bool *__i1, bool __i2)`
- `template<typename _ITp > bool std::atomic_compare_exchange_strong (volatile __atomic_base< _ITp > *__a, _ITp *__i1, _ITp __i2)`
- `template<typename _ITp > bool std::atomic_compare_exchange_strong (__atomic_base< _ITp > *__a, _ITp *__i1, _ITp __i2)`
- `template<typename _ITp > bool std::atomic_compare_exchange_strong_explicit (__atomic_base< _ITp > *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2)`
- `template<typename _ITp > bool std::atomic_compare_exchange_strong_explicit (volatile __atomic_base< _ITp > *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2)`
- `bool std::atomic_compare_exchange_strong_explicit (atomic_address *__a, void **__v1, void *__v2, memory_order __m1, memory_order __m2)`
- `bool std::atomic_compare_exchange_strong_explicit (volatile atomic_address *__a, void **__v1, void *__v2, memory_order __m1, memory_order __m2)`
- `bool std::atomic_compare_exchange_strong_explicit (atomic_bool *__a, bool *__i1, bool __i2, memory_order __m1, memory_order __m2)`
- `bool std::atomic_compare_exchange_strong_explicit (volatile atomic_bool *__a, bool *__i1, bool __i2, memory_order __m1, memory_order __m2)`
- `template<typename _ITp > bool std::atomic_compare_exchange_weak (__atomic_base< _ITp > *__a, _ITp *__i1, _ITp __i2)`
- `template<typename _ITp > bool std::atomic_compare_exchange_weak (volatile __atomic_base< _ITp > *__a, _ITp *__i1, _ITp __i2)`
- `bool std::atomic_compare_exchange_weak (atomic_bool *__a, bool *__i1, bool __i2)`
- `bool std::atomic_compare_exchange_weak (atomic_address *__a, void **__v1, void *__v2)`
- `bool std::atomic_compare_exchange_weak (volatile atomic_bool *__a, bool *__i1, bool __i2)`

- `bool std::atomic_compare_exchange_weak` (volatile atomic_address *__a, void **__v1, void *__v2)
- `template<typename _ITp >`
`bool std::atomic_compare_exchange_weak_explicit` (volatile __atomic_base< _ITp > *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2)
- `template<typename _ITp >`
`bool std::atomic_compare_exchange_weak_explicit` (__atomic_base< _ITp > *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2)
- `bool std::atomic_compare_exchange_weak_explicit` (volatile atomic_address *__a, void **__v1, void *__v2, memory_order __m1, memory_order __m2)
- `bool std::atomic_compare_exchange_weak_explicit` (atomic_bool *__a, bool *__i1, bool __i2, memory_order __m1, memory_order __m2)
- `bool std::atomic_compare_exchange_weak_explicit` (atomic_address *__a, void **__v1, void *__v2, memory_order __m1, memory_order __m2)
- `bool std::atomic_compare_exchange_weak_explicit` (volatile atomic_bool *__a, bool *__i1, bool __i2, memory_order __m1, memory_order __m2)
- `template<typename _ITp >`
`_ITp std::atomic_exchange` (__atomic_base< _ITp > *__a, _ITp __i)
- `template<typename _ITp >`
`_ITp std::atomic_exchange` (volatile __atomic_base< _ITp > *__a, _ITp __i)
- `void * std::atomic_exchange` (atomic_address *__a, void *__v)
- `void * std::atomic_exchange` (volatile atomic_address *__a, void *__v)
- `bool std::atomic_exchange` (atomic_bool *__a, bool __i)
- `bool std::atomic_exchange` (volatile atomic_bool *__a, bool __i)
- `void * std::atomic_exchange_explicit` (atomic_address *__a, void *__v, memory_order __m)
- `template<typename _ITp >`
`_ITp std::atomic_exchange_explicit` (__atomic_base< _ITp > *__a, _ITp __i, memory_order __m)
- `template<typename _ITp >`
`_ITp std::atomic_exchange_explicit` (volatile __atomic_base< _ITp > *__a, _ITp __i, memory_order __m)
- `void * std::atomic_exchange_explicit` (volatile atomic_address *__a, void *__v, memory_order __m)
- `bool std::atomic_exchange_explicit` (atomic_bool *__a, bool __i, memory_order __m)
- `bool std::atomic_exchange_explicit` (volatile atomic_bool *__a, bool __i, memory_order __m)
- `void * std::atomic_fetch_add` (atomic_address *__a, ptrdiff_t __d)
- `void * std::atomic_fetch_add` (volatile atomic_address *__a, ptrdiff_t __d)
- `template<typename _ITp >`
`_ITp std::atomic_fetch_add` (__atomic_base< _ITp > *__a, _ITp __i)

- `template<typename _ITp >`
`_ITp std::atomic_fetch_add (volatile __atomic_base< _ITp > *__a, _ITp __i)`
- `void * std::atomic_fetch_add_explicit (atomic_address *__a, ptrdiff_t __d,`
`memory_order __m)`
- `void * std::atomic_fetch_add_explicit (volatile atomic_address *__a, ptrdiff_t`
`__d, memory_order __m)`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_add_explicit (__atomic_base< _ITp > *__a, _ITp __i,`
`memory_order __m)`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_add_explicit (volatile __atomic_base< _ITp > *__a,`
`_ITp __i, memory_order __m)`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_and (__atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_and (volatile __atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_and_explicit (__atomic_base< _ITp > *__a, _ITp __i,`
`memory_order __m)`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_and_explicit (volatile __atomic_base< _ITp > *__a,`
`_ITp __i, memory_order __m)`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_or (volatile __atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_or (__atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_or_explicit (__atomic_base< _ITp > *__a, _ITp __i,`
`memory_order __m)`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_or_explicit (volatile __atomic_base< _ITp > *__a, _-`
`_ITp __i, memory_order __m)`
- `void * std::atomic_fetch_sub (atomic_address *__a, ptrdiff_t __d)`
- `void * std::atomic_fetch_sub (volatile atomic_address *__a, ptrdiff_t __d)`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_sub (__atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_sub (volatile __atomic_base< _ITp > *__a, _ITp __i)`
- `void * std::atomic_fetch_sub_explicit (atomic_address *__a, ptrdiff_t __d,`
`memory_order __m)`
- `void * std::atomic_fetch_sub_explicit (volatile atomic_address *__a, ptrdiff_t`
`__d, memory_order __m)`

- `template<typename _ITp >`
`_ITp std::atomic_fetch_sub_explicit (__atomic_base< _ITp > *__a, _ITp __i,`
`memory_order __m)`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_sub_explicit (volatile __atomic_base< _ITp > *__a,`
`_ITp __i, memory_order __m)`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_xor (volatile __atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_xor (__atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_xor_explicit (volatile __atomic_base< _ITp > *__a,`
`_ITp __i, memory_order __m)`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_xor_explicit (__atomic_base< _ITp > *__a, _ITp __i,`
`memory_order __m)`
- `void std::atomic_flag_clear (atomic_flag *__a)`
- `void std::atomic_flag_clear (volatile atomic_flag *__a)`
- `void std::atomic_flag_clear_explicit (volatile atomic_flag *__a, memory_`
`order __m)`
- `void std::atomic_flag_clear_explicit (atomic_flag *__a, memory_order __m)`
- `bool std::atomic_flag_test_and_set (atomic_flag *__a)`
- `bool std::atomic_flag_test_and_set (volatile atomic_flag *__a)`
- `bool std::atomic_flag_test_and_set_explicit (volatile atomic_flag *__a,`
`memory_order __m)`
- `bool std::atomic_flag_test_and_set_explicit (atomic_flag *__a, memory_`
`order __m)`
- `template<typename _ITp >`
`void std::atomic_init (__atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`
`void std::atomic_init (volatile __atomic_base< _ITp > *__a, _ITp __i)`
- `void std::atomic_init (atomic_bool *__a, bool __b)`
- `void std::atomic_init (atomic_address *__a, void *__v)`
- `void std::atomic_init (volatile atomic_address *__a, void *__v)`
- `void std::atomic_init (volatile atomic_bool *__a, bool __b)`
- `template<typename _ITp >`
`bool std::atomic_is_lock_free (const __atomic_base< _ITp > *__a)`
- `template<typename _ITp >`
`bool std::atomic_is_lock_free (const volatile __atomic_base< _ITp > *__a)`
- `bool std::atomic_is_lock_free (const volatile atomic_address *__a)`
- `bool std::atomic_is_lock_free (const atomic_address *__a)`
- `bool std::atomic_is_lock_free (const volatile atomic_bool *__a)`
- `bool std::atomic_is_lock_free (const atomic_bool *__a)`

- `template<typename _ITp >`
`_ITp std::atomic_load (const volatile __atomic_base< _ITp > *__a)`
- `void * std::atomic_load (const atomic_address *__a)`
- `void * std::atomic_load (const volatile atomic_address *__a)`
- `template<typename _ITp >`
`_ITp std::atomic_load (const __atomic_base< _ITp > *__a)`
- `bool std::atomic_load (const atomic_bool *__a)`
- `bool std::atomic_load (const volatile atomic_bool *__a)`
- `void * std::atomic_load_explicit (const atomic_address *__a, memory_order __m)`
- `template<typename _ITp >`
`_ITp std::atomic_load_explicit (const __atomic_base< _ITp > *__a, memory_order __m)`
- `template<typename _ITp >`
`_ITp std::atomic_load_explicit (const volatile __atomic_base< _ITp > *__a, memory_order __m)`
- `void * std::atomic_load_explicit (const volatile atomic_address *__a, memory_order __m)`
- `bool std::atomic_load_explicit (const atomic_bool *__a, memory_order __m)`
- `bool std::atomic_load_explicit (const volatile atomic_bool *__a, memory_order __m)`
- `void std::atomic_store (atomic_bool *__a, bool __i)`
- `void std::atomic_store (volatile atomic_address *__a, void *__v)`
- `template<typename _ITp >`
`void std::atomic_store (volatile __atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`
`void std::atomic_store (__atomic_base< _ITp > *__a, _ITp __i)`
- `void std::atomic_store (volatile atomic_bool *__a, bool __i)`
- `void std::atomic_store (atomic_address *__a, void *__v)`
- `template<typename _ITp >`
`void std::atomic_store_explicit (volatile __atomic_base< _ITp > *__a, _ITp __i, memory_order __m)`
- `void std::atomic_store_explicit (volatile atomic_bool *__a, bool __i, memory_order __m)`
- `void std::atomic_store_explicit (atomic_bool *__a, bool __i, memory_order __m)`
- `template<typename _ITp >`
`void std::atomic_store_explicit (__atomic_base< _ITp > *__a, _ITp __i, memory_order __m)`
- `void std::atomic_store_explicit (volatile atomic_address *__a, void *__v, memory_order __m)`
- `void std::atomic_store_explicit (atomic_address *__a, void *__v, memory_order __m)`

- `bool std::atomic< _Tp * >::compare_exchange_strong (_Tp * &, _Tp *, memory_order, memory_order) volatile`
- `bool std::atomic< _Tp * >::compare_exchange_strong (_Tp * &, _Tp *, memory_order=memory_order_seq_cst) volatile`
- `bool std::atomic< _Tp * >::compare_exchange_strong (_Tp * &, _Tp *, memory_order, memory_order)`
- `bool std::atomic< _Tp * >::compare_exchange_strong (_Tp * &, _Tp *, memory_order=memory_order_seq_cst)`
- `bool std::atomic< _Tp * >::compare_exchange_weak (_Tp * &, _Tp *, memory_order, memory_order)`
- `bool std::atomic< _Tp * >::compare_exchange_weak (_Tp * &, _Tp *, memory_order, memory_order) volatile`
- `bool std::atomic< _Tp * >::compare_exchange_weak (_Tp * &, _Tp *, memory_order=memory_order_seq_cst)`
- `bool std::atomic< _Tp * >::compare_exchange_weak (_Tp * &, _Tp *, memory_order=memory_order_seq_cst) volatile`
- `_Tp * std::atomic< _Tp * >::exchange (_Tp *, memory_order=memory_order_seq_cst)`
- `_Tp * std::atomic< _Tp * >::exchange (_Tp *, memory_order=memory_order_seq_cst) volatile`
- `_Tp * std::atomic< _Tp * >::fetch_add (ptrdiff_t, memory_order=memory_order_seq_cst)`
- `_Tp * std::atomic< _Tp * >::fetch_add (ptrdiff_t, memory_order=memory_order_seq_cst) volatile`
- `_Tp * std::atomic< _Tp * >::fetch_sub (ptrdiff_t, memory_order=memory_order_seq_cst)`
- `_Tp * std::atomic< _Tp * >::fetch_sub (ptrdiff_t, memory_order=memory_order_seq_cst) volatile`
- `template<typename _Tp >`
`_Tp std::kill_dependency (_Tp __y)`
- `_Tp * std::atomic< _Tp * >::load (memory_order=memory_order_seq_cst) const`
- `_Tp * std::atomic< _Tp * >::load (memory_order=memory_order_seq_cst) const volatile`

2.33.1 Detailed Description

Components for performing atomic operations.

2.33.2 Define Documentation

2.33.2.1 #define _GLIBCXX_ATOMIC_PROPERTY

Lock-free Property.

Definition at line 142 of file atomic_base.h.

2.33.3 Typedef Documentation

2.33.3.1 typedef __atomic_base<char> std::atomic_char

atomic_char

Definition at line 166 of file atomic_base.h.

2.33.3.2 typedef __atomic_base<char16_t> std::atomic_char16_t

atomic_char16_t

Definition at line 202 of file atomic_base.h.

2.33.3.3 typedef __atomic_base< char32_t > std::atomic_char32_t

atomic_char32_t

Definition at line 205 of file atomic_base.h.

2.33.3.4 typedef __atomic_base<int> std::atomic_int

atomic_int

Definition at line 181 of file atomic_base.h.

2.33.3.5 typedef __atomic_base<int_fast16_t> std::atomic_int_fast16_t

atomic_int_fast16_t

Definition at line 243 of file atomic_base.h.

2.33.3.6 typedef __atomic_base<int_fast32_t> std::atomic_int_fast32_t

atomic_int_fast32_t

Definition at line 249 of file atomic_base.h.

2.33.3.7 typedef __atomic_base<int_fast64_t> std::atomic_int_fast64_t

atomic_int_fast64_t

Definition at line 255 of file atomic_base.h.

2.33.3.8 typedef __atomic_base<int_fast8_t> std::atomic_int_fast8_t

atomic_int_fast8_t

Definition at line 237 of file atomic_base.h.

2.33.3.9 typedef __atomic_base<int_least16_t> std::atomic_int_least16_t

atomic_int_least16_t

Definition at line 218 of file atomic_base.h.

2.33.3.10 typedef __atomic_base<int_least32_t> std::atomic_int_least32_t

atomic_int_least32_t

Definition at line 224 of file atomic_base.h.

2.33.3.11 typedef __atomic_base<int_least64_t> std::atomic_int_least64_t

atomic_int_least64_t

Definition at line 230 of file atomic_base.h.

2.33.3.12 typedef __atomic_base<int_least8_t> std::atomic_int_least8_t

atomic_int_least8_t

Definition at line 212 of file atomic_base.h.

2.33.3.13 typedef __atomic_base<intmax_t> std::atomic_intmax_t

atomic_intmax_t

Definition at line 271 of file atomic_base.h.

2.33.3.14 typedef __atomic_base<intptr_t> std::atomic_intptr_t

atomic_intptr_t

Definition at line 262 of file atomic_base.h.

2.33.3.15 typedef __atomic_base<long long> std::atomic_llong

atomic_llong

Definition at line 193 of file atomic_base.h.

2.33.3.16 typedef __atomic_base<long> std::atomic_long

atomic_long

Definition at line 187 of file atomic_base.h.

2.33.3.17 typedef __atomic_base<ptrdiff_t> std::atomic_ptrdiff_t

atomic_ptrdiff_t

Definition at line 277 of file atomic_base.h.

2.33.3.18 typedef __atomic_base<signed char> std::atomic_schar

atomic_schar

Definition at line 169 of file atomic_base.h.

2.33.3.19 typedef __atomic_base<short> std::atomic_short

atomic_short

Definition at line 175 of file atomic_base.h.

2.33.3.20 typedef __atomic_base<size_t> std::atomic_size_t

atomic_size_t

Definition at line 268 of file atomic_base.h.

2.33.3.21 typedef __atomic_base<unsigned char> std::atomic_uchar

atomic_uchar

Definition at line 172 of file atomic_base.h.

2.33.3.22 typedef __atomic_base<unsigned int> std::atomic_uint

atomic_uint

Definition at line 184 of file atomic_base.h.

2.33.3.23 typedef __atomic_base<uint_fast16_t> std::atomic_uint_fast16_t

atomic_uint_fast16_t

Definition at line 246 of file atomic_base.h.

2.33.3.24 `typedef __atomic_base<uint_fast32_t> std::atomic_uint_fast32_t`

`atomic_uint_fast32_t`

Definition at line 252 of file `atomic_base.h`.

2.33.3.25 `typedef __atomic_base<uint_fast64_t> std::atomic_uint_fast64_t`

`atomic_uint_fast64_t`

Definition at line 258 of file `atomic_base.h`.

2.33.3.26 `typedef __atomic_base<uint_fast8_t> std::atomic_uint_fast8_t`

`atomic_uint_fast8_t`

Definition at line 240 of file `atomic_base.h`.

2.33.3.27 `typedef __atomic_base<uint_least16_t> std::atomic_uint_least16_t`

`atomic_uint_least16_t`

Definition at line 221 of file `atomic_base.h`.

2.33.3.28 `typedef __atomic_base<uint_least32_t> std::atomic_uint_least32_t`

`atomic_uint_least32_t`

Definition at line 227 of file `atomic_base.h`.

2.33.3.29 `typedef __atomic_base<uint_least64_t> std::atomic_uint_least64_t`

`atomic_uint_least64_t`

Definition at line 233 of file `atomic_base.h`.

2.33.3.30 typedef __atomic_base<uint_least8_t> std::atomic_uint_least8_t

atomic_uint_least8_t

Definition at line 215 of file atomic_base.h.

2.33.3.31 typedef __atomic_base<uintmax_t> std::atomic_uintmax_t

atomic_uintmax_t

Definition at line 274 of file atomic_base.h.

2.33.3.32 typedef __atomic_base<uintptr_t> std::atomic_uintptr_t

atomic_uintptr_t

Definition at line 265 of file atomic_base.h.

2.33.3.33 typedef __atomic_base<unsigned long long> std::atomic_ullong

atomic_ullong

Definition at line 196 of file atomic_base.h.

2.33.3.34 typedef __atomic_base<unsigned long> std::atomic_ulong

atomic_ulong

Definition at line 190 of file atomic_base.h.

2.33.3.35 typedef __atomic_base<unsigned short> std::atomic_ushort

atomic_ushort

Definition at line 178 of file atomic_base.h.

2.33.3.36 typedef __atomic_base<wchar_t> std::atomic_wchar_t

atomic_wchar_t

Definition at line 199 of file atomic_base.h.

2.33.3.37 typedef enum std::memory_order std::memory_order

Enumeration for memory_order.

2.33.4 Enumeration Type Documentation**2.33.4.1 enum std::memory_order**

Enumeration for memory_order.

Definition at line 51 of file atomic_base.h.

2.33.5 Function Documentation**2.33.5.1 template<typename _Tp > _Tp std::kill_dependency (_Tp __y)
[inline]**

kill_dependency

Definition at line 74 of file atomic_base.h.

2.34 Hashes

Collaboration diagram for Hashes:



Classes

- struct `std::hash<_Tp>`
Primary class template hash.
- struct `std::hash<_Tp*>`
Partial specializations for pointer types.

Defines

- `#define _Cxx_hashtable_define_trivial_hash(_Tp)`

2.34.1 Detailed Description

Hashing functors taking a variable type and returning a `std::size_t`.

2.35 Locales

Classes

- class `std::codecvt<_InternT, _ExternT, _StateT>`
Primary class template codecvt.
NB: Generic, mostly useless implementation.
- class `std::ctype<_CharT>`

Primary class template ctype facet.

This template class defines classification and conversion functions for character sets.

It wraps ctype functionality. Ctype gets used by streams for many I/O operations.

- class `std::ctype< char >`

The `ctype<char>` specialization.

This class defines classification and conversion functions for the char type. It gets

used by char streams for many I/O operations. The char specialization provides a

number of optimizations as well.

- class `std::ctype< wchar_t >`

The `ctype<wchar_t>` specialization.

This class defines classification and conversion functions for the wchar_t type. It

gets used by wchar_t streams for many I/O operations. The wchar_t specialization

provides a number of optimizations as well.

- class `std::locale`

Container class for localization functionality.

The locale class is first a class wrapper for C library locales. It is also an exten-

sible container for user-defined localization. A locale is a collection of facets that

implement various localization features such as money, time, and number printing.

- class `std::locale::facet`

Localization functionality base class.

The facet class is the base class for a localization feature, such as money, time, and

number printing. It provides common support for facets and reference management.

- class `std::locale::id`

Facet ID class.

The ID class provides facets with an index used to identify them. Every facet class

must define a public static member `locale::id`, or be derived from a facet that provides

this member; otherwise the facet cannot be used in a locale. The `locale::id` ensures

that each class type gets a unique identifier.

- class `std::messages< _CharT >`

Primary class template messages.

This facet encapsulates the code to retrieve messages from message catalogs. The

only thing defined by the standard for this facet is the interface. All underlying func-

tionality is implementation-defined.

- struct `std::messages_base`

Messages facet base class providing catalog typedef.

- class `std::money_base`

Money format ordering data.

This class contains an ordered array of 4 fields to represent the pattern for formatting a money amount. Each field may contain one entry from the part enum. symbol, sign, and value must be present and the remaining field must contain either none or space.

- class `std::money_get< _CharT, _InIter >`

Primary class template `money_get`.

This facet encapsulates the code to parse and return a monetary amount from a string.

- class `std::money_put< _CharT, _OutIter >`

Primary class template `money_put`.

This facet encapsulates the code to format and output a monetary amount.

- class `std::moneypunct< _CharT, _Intl >`

Primary class template `moneypunct`.

This facet encapsulates the punctuation, grouping and other formatting features of money amount string representations.

- class `std::num_get< _CharT, _InIter >`

Primary class template `num_get`.

This facet encapsulates the code to parse and return a number from a string. It is used by the istream numeric extraction operators.

- class `std::num_put< _CharT, _OutIter >`

Primary class template `num_put`.

This facet encapsulates the code to convert a number to a string. It is used by the ostream numeric insertion operators.

- class `std::numpunct< _CharT >`

Primary class template `numpunct`.

This facet stores several pieces of information related to printing and scanning numbers, such as the decimal point character. It takes a template parameter specifying the char type. The numpunct facet is used by streams for many I/O operations involving numbers.

- class `std::time_base`

Time format ordering data.

This class provides an enum representing different orderings of time: day, month, and year.

- class `std::time_get< _CharT, _InIter >`

Primary class template `time_get`.

This facet encapsulates the code to parse and return a date or time from a string. It is used by the istream numeric extraction operators.

- class [std::time_put<_CharT, _OutIter>](#)

Primary class template [time_put](#).

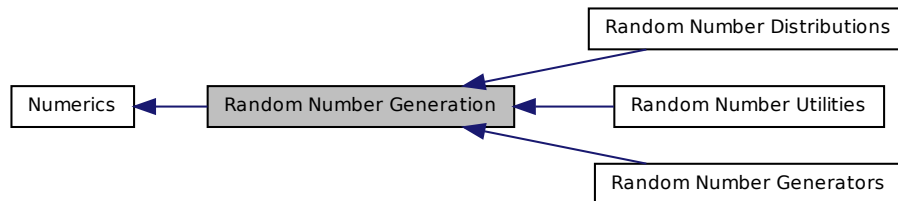
This facet encapsulates the code to format and output dates and times according to formats used by [strftime\(\)](#).

2.35.1 Detailed Description

Classes and functions for internationalization and localization.

2.36 Random Number Generation

Collaboration diagram for Random Number Generation:



Namespaces

- namespace [std::__detail](#)

Modules

- [Random Number Generators](#)
- [Random Number Distributions](#)
- [Random Number Utilities](#)

Functions

- template<typename [_RealType](#) , size_t [__bits](#), typename [_UniformRandomNumberGenerator](#) >
[_RealType](#) [std::generate_canonical](#) ([_UniformRandomNumberGenerator](#) &[__g](#))

2.36.1 Detailed Description

A facility for generating random numbers on selected distributions.

2.36.2 Function Documentation

2.36.2.1 `template<typename _RealType , size_t __bits, typename
_UniformRandomNumberGenerator > _RealType
std::generate_canonical (_UniformRandomNumberGenerator & __g
)`

A function template for converting the output of a (integral) uniform random number generator to a floating point result in the range [0-1).

Definition at line 2808 of file random.tcc.

References `std::log()`, and `std::min()`.

2.37 Regular Expressions

Classes

- class `std::basic_regex< _Ch_type, _Rx_traits >`
- class `std::match_results< _Bi_iter, _Allocator >`
The results of a match or search operation.
- class `std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >`
- class `std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >`
- struct `std::regex_traits< _Ch_type >`
Describes aspects of a regular expression.
- class `std::sub_match< _BiIter >`

Typedefs

- typedef `match_results< const char * >` `std::cmatch`
- typedef `regex_iterator< const char * >` `std::cregex_iterator`
- typedef `regex_token_iterator< const char * >` `std::cregex_token_iterator`
- typedef `sub_match< const char * >` `std::csub_match`
- typedef `basic_regex< char >` `std::regex`
- typedef `match_results< string::const_iterator >` `std::smatch`
- typedef `regex_iterator< string::const_iterator >` `std::sregex_iterator`

- `typedef regex_token_iterator< string::const_iterator > std::sregex_token_iterator`
- `typedef sub_match< string::const_iterator > std::ssub_match`
- `typedef match_results< const wchar_t * > std::wcmatch`
- `typedef regex_iterator< const wchar_t * > std::wcregex_iterator`
- `typedef regex_token_iterator< const wchar_t * > std::wcregex_token_iterator`
- `typedef sub_match< const wchar_t * > std::wcsub_match`
- `typedef basic_regex< wchar_t > std::wregex`
- `typedef match_results< wstring::const_iterator > std::wsmatch`
- `typedef regex_iterator< wstring::const_iterator > std::wsregex_iterator`
- `typedef regex_token_iterator< wstring::const_iterator > std::wsregex_token_iterator`
- `typedef sub_match< wstring::const_iterator > std::wssub_match`

Functions

- `template<typename _Bi_iter >
const sub_match< _Bi_iter > & std::__unmatched_sub ()`
- `bool std::regex_traits::isctype (_Ch_type __c, char_class_type __f) const`
- `template<typename _Biter >
bool std::operator!= (const sub_match< _Biter > &__lhs, const sub_match< _Biter > &__rhs)`
- `template<typename _Bi_iter >
bool std::operator!= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >
bool std::operator!= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >
bool std::operator!= (const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >
bool std::operator!= (typename iterator_traits< _Bi_iter >::value_type const * - __lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Allocator >
bool std::operator!= (const match_results< _Bi_iter, _Allocator > &__m1, const match_results< _Bi_iter, _Allocator > &__m2)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >
bool std::operator!= (const sub_match< _Bi_iter > &__lhs, const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__rhs)`

- `template<typename _Bi_iter >`
`bool std::operator!= (const sub_match< _Bi_iter > &__lhs, typename iterator_`
`traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bilter >`
`bool std::operator< (const sub_match< _Bilter > &__lhs, const sub_match<`
`_Bilter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool std::operator< (const sub_match< _Bi_iter > &__lhs, const basic_string<`
`typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__`
`_rhs)`
- `template<typename _Bi_iter >`
`bool std::operator< (typename iterator_traits< _Bi_iter >::value_type const`
`&__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator< (const basic_string< typename iterator_traits< _Bi_iter`
`>::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter >`
`&__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_`
`traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator< (typename iterator_traits< _Bi_iter >::value_type const *_`
`__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_`
`traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Ch_type, typename _Ch_traits, typename _Bi_iter >`
`basic_ostream< _Ch_type, _Ch_traits > & std::operator<< (basic_ostream<`
`_Ch_type, _Ch_traits > &__os, const sub_match< _Bi_iter > &__m)`
- `template<typename _Bi_iter >`
`bool std::operator<= (typename iterator_traits< _Bi_iter >::value_type const`
`&__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool std::operator<= (const sub_match< _Bi_iter > &__lhs, const basic_`
`string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_`
`alloc > &__rhs)`
- `template<typename _Bilter >`
`bool std::operator<= (const sub_match< _Bilter > &__lhs, const sub_match<`
`_Bilter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator<= (const sub_match< _Bi_iter > &__lhs, typename`
`iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator<= (typename iterator_traits< _Bi_iter >::value_type const`
`*__lhs, const sub_match< _Bi_iter > &__rhs)`

- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator<= (const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator<= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator== (const sub_match< _Bi_iter > &__lhs, const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator== (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator== (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator== (const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, typename _Allocator >`
`bool std::operator== (const match_results< _Bi_iter, _Allocator > &__m1, const match_results< _Bi_iter, _Allocator > &__m2)`
- `template<typename _BiIter >`
`bool std::operator== (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator> (const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool std::operator> (const sub_match< _Bi_iter > &__lhs, const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__rhs)`

- `template<typename _Bi_iter >`
`bool std::operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_`
`traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator> (typename iterator_traits< _Bi_iter >::value_type const *__`
`_lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator> (typename iterator_traits< _Bi_iter >::value_type const`
`&__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _BiIter >`
`bool std::operator> (const sub_match< _BiIter > &__lhs, const sub_match<`
`_BiIter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool std::operator>= (const sub_match< _Bi_iter > &__lhs, const basic_`
`string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_`
`alloc > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator>= (const sub_match< _Bi_iter > &__lhs, typename`
`iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator>= (const basic_string< typename iterator_traits< _Bi_iter`
`>::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter >`
`&__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator>= (typename iterator_traits< _Bi_iter >::value_type const`
`&__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator>= (typename iterator_traits< _Bi_iter >::value_type const`
`*__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _BiIter >`
`bool std::operator>= (const sub_match< _BiIter > &__lhs, const sub_match<`
`_BiIter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator>= (const sub_match< _Bi_iter > &__lhs, typename`
`iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter, typename _Allocator >`
`void std::swap (match_results< _Bi_iter, _Allocator > &__lhs, match_results<`
`_Bi_iter, _Allocator > &__rhs)`
- `template<typename _Ch_type, typename _Rx_traits >`
`void std::swap (basic_regex< _Ch_type, _Rx_traits > &__lhs, basic_regex< _`
`Ch_type, _Rx_traits > &__rhs)`
- `int std::regex_traits::value (_Ch_type __ch, int __radix) const`

Matching, Searching, and Replacing

- `template<typename _Bi_iter, typename _Allocator, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_match (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _`
`Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_`
`constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_match (_Bi_iter __first, _Bi_iter __last, const basic_`
`regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __`
`flags=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Allocator, typename _Rx_traits >`
`bool std::regex_match (const _Ch_type *__s, match_results< const _Ch_type`
`*, _Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re,`
`regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Allocator, typename _Ch_type,`
`typename _Rx_traits >`
`bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc`
`> &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _`
`Ch_alloc >::const_iterator, _Allocator > &__m, const basic_regex< _Ch_`
`type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_`
`constants::match_default)`
- `template<typename _Ch_type, class _Rx_traits >`
`bool std::regex_match (const _Ch_type *__s, const basic_regex< _Ch_`
`type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_`
`constants::match_default)`
- `template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _Rx_`
`traits >`
`bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Str_`
`allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_`
`constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Allocator, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_search (_Bi_iter __first, _Bi_iter __last, match_results< _Bi_`
`iter, _Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re,`
`regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_search (_Bi_iter __first, _Bi_iter __last, const basic_`
`regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __`
`flags=regex_constants::match_default)`
- `template<typename _Ch_type, class _Allocator, class _Rx_traits >`
`bool std::regex_search (const _Ch_type *__s, match_results< const _Ch_type *`
`, _Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_`
`constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Rx_traits >`
`bool std::regex_search (const _Ch_type *__s, const basic_regex< _Ch_`

```
type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_
constants::match_default)
```

- `template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _String_allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Allocator, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type >`
`_Out_iter std::regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string< _Ch_type > &__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type >`
`basic_string< _Ch_type > std::regex_replace (const basic_string< _Ch_type > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string< _Ch_type > &__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)`

2.37.1 Detailed Description

A facility for performing regular expression pattern matching.

2.37.2 Typedef Documentation

2.37.2.1 `typedef regex_token_iterator<const char*> std::cregex_token_iterator`

Token iterator for C-style NULL-terminated strings.

Definition at line 2415 of file `regex.h`.

2.37.2.2 `typedef sub_match<const char*> std::csub_match`

Standard regex submatch over a C-style null-terminated string.

Definition at line 847 of file regex.h.

2.37.2.3 `typedef basic_regex<char> std::regex`

Standard regular expressions.

Definition at line 722 of file regex.h.

2.37.2.4 `typedef regex_token_iterator<string::const_iterator> std::sregex_token_iterator`

Token iterator for standard strings.

Definition at line 2417 of file regex.h.

2.37.2.5 `typedef sub_match<string::const_iterator> std::ssub_match`

Standard regex submatch over a standard string.

Definition at line 849 of file regex.h.

2.37.2.6 `typedef regex_token_iterator<const wchar_t*> std::wregex_token_iterator`

Token iterator for C-style NULL-terminated wide strings.

Definition at line 2420 of file regex.h.

2.37.2.7 `typedef sub_match<const wchar_t*> std::wsub_match`

Regex submatch over a C-style null-terminated wide string.

Definition at line 852 of file regex.h.

2.37.2.8 `typedef basic_regex<wchar_t> std::wregex`

Standard wide-character regular expressions.

Definition at line 725 of file regex.h.

2.37.2.9 `typedef regex_token_iterator<wstring::const_iterator>
std::wsregex_token_iterator`

Token iterator for standard wide-character strings.

Definition at line 2422 of file regex.h.

2.37.2.10 `typedef sub_match<wstring::const_iterator> std::wssub_match`

Regex submatch over a standard wide string.

Definition at line 854 of file regex.h.

2.37.3 Function Documentation

2.37.3.1 `template<typename _Ch_type > bool std::regex_traits<_Ch_type
>::isctype (_Ch_type __c, char_class_type __f) const
[inherited]`

Determines if *c* is a member of an identified class.

Parameters

c a character.

f a class type (as returned from lookup_classname).

Returns

true if the character *c* is a member of the classification represented by *f*, false otherwise.

Exceptions

[*std::bad_cast*](#) if the current locale does not have a ctype facet.

Definition at line 284 of file regex.h.

References `std::__ctype_abstract_base<_CharT >::is()`, `std::regex_traits<_Ch_type >::lookup_classname()`, `std::use_facet()`, and `std::__ctype_abstract_base<_CharT >::widen()`.


```
2.37.3.2 template<typename _Bilter > bool std::operator!= ( const  
    sub_match< _Bilter > & __lhs, const sub_match< _Bilter > &  
    __rhs ) [inline]
```

Tests the inequivalence of two regular expression submatches.

Parameters

lhs First regular expression submatch.

rhs Second regular expression submatch.

Returns

true if *lhs* is not equivalent to *rhs*, false otherwise.

Definition at line 879 of file regex.h.

References std::sub_match< _Bilter >::compare().

```
2.37.3.3 template<typename _Bi_iter > bool std::operator!= ( typename  
    iterator_traits< _Bi_iter >::value_type const & __lhs, const  
    sub_match< _Bi_iter > & __rhs ) [inline]
```

Tests the inequivalence of a string and a regular expression submatch.

Parameters

lhs A string.

rhs A regular expression submatch.

Returns

true if *lhs* is not equivalent to *rhs*, false otherwise.

Definition at line 1268 of file regex.h.

References std::sub_match< _Bilter >::str().

```
2.37.3.4 template<typename _Bi_iter > bool std::operator!= ( const  
    sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter  
    >::value_type const & __rhs ) [inline]
```

Tests the inequivalence of a regular expression submatch and a string.

Parameters

lhs A regular expression submatch.

rhs A const string reference.

Returns

true if *lhs* is not equivalent to *rhs*, false otherwise.

Definition at line 1342 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

2.37.3.5 `template<typename _Bi_iter , typename _Ch_traits , typename
_Ch_alloc > bool std::operator!=(const basic_string< typename
iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &
__lhs, const sub_match< _Bi_iter > & __rhs) [inline]`

Tests the inequivalence of a string and a regular expression submatch.

Parameters

lhs A string.

rhs A regular expression submatch.

Returns

true if *lhs* is not equivalent to *rhs*, false otherwise.

Definition at line 955 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

2.37.3.6 `template<typename _Bi_iter > bool std::operator!=(typename
iterator_traits< _Bi_iter >::value_type const * __lhs, const
sub_match< _Bi_iter > & __rhs) [inline]`

Tests the inequivalence of an iterator value and a regular expression submatch.

Parameters

lhs A regular expression submatch.

rhs A string.

Returns

true if *lhs* is not equivalent to *rhs*, false otherwise.

Definition at line 1120 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

2.37.3.7 `template<typename _Bi_iter, class _Allocator> bool std::operator!=(
const match_results<_Bi_iter, _Allocator> & __m1, const
match_results<_Bi_iter, _Allocator> & __m2) [inline]`

Compares two [match_results](#) for inequality.

Returns

true if the two objects do not refer to the same match, false otherwise.

Definition at line 1779 of file regex.h.

2.37.3.8 `template<typename _Bi_iter> bool std::operator!=(const
sub_match<_Bi_iter> & __lhs, typename iterator_traits<_Bi_iter>
::value_type const * __rhs) [inline]`

Tests the inequivalence of a regular expression submatch and a string.

Parameters

lhs A regular expression submatch.

rhs A pointer to a string.

Returns

true if *lhs* is not equivalent to *rhs*, false otherwise.

Definition at line 1194 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

2.37.3.9 `template<typename _Bi_iter, typename _Ch_traits, typename
_Ch_alloc> bool std::operator!=(const sub_match<_Bi_iter>
& __lhs, const basic_string<typename iterator_traits<_Bi_iter>
::value_type, _Ch_traits, _Ch_alloc> & __rhs) [inline]`

Tests the inequivalence of a regular expression submatch and a string.

Parameters

lhs A regular expression submatch.

rhs A string.

Returns

true if *lhs* is not equivalent to *rhs*, false otherwise.

Definition at line 1036 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

2.37.3.10 `template<typename _BiIter > bool std::operator< (const
sub_match<_BiIter> & __lhs, const sub_match<_BiIter> &
__rhs) [inline]`

Tests the ordering of two regular expression submatches.

Parameters

lhs First regular expression submatch.

rhs Second regular expression submatch.

Returns

true if *lhs* precedes *rhs*, false otherwise.

Definition at line 891 of file regex.h.

2.37.3.11 `template<typename _Bi_iter , class _Ch_traits , class _Ch_alloc >
bool std::operator< (const sub_match<_Bi_iter> & __lhs, const
basic_string< typename iterator_traits<_Bi_iter>::value_type,
_Ch_traits, _Ch_alloc > & __rhs) [inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

lhs A regular expression submatch.

rhs A string.

Returns

true if *lhs* precedes *rhs*, false otherwise.

Definition at line 1050 of file regex.h.

2.37.3.12 `template<typename _Bi_iter > bool std::operator< (typename
iterator_traits<_Bi_iter>::value_type const & __lhs, const
sub_match<_Bi_iter> & __rhs) [inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

lhs A string.

rhs A regular expression submatch.

Returns

true if *lhs* precedes *rhs*, false otherwise.

Definition at line 1280 of file regex.h.

2.37.3.13 `template<typename _Bi_iter , typename _Ch_traits , typename
_Ch_alloc > bool std::operator< (const basic_string< typename
iterator_traits<_Bi_iter>::value_type, _Ch_traits, _Ch_alloc > &
__lhs, const sub_match<_Bi_iter> & __rhs) [inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

lhs A string.

rhs A regular expression submatch.

Returns

true if *lhs* precedes *rhs*, false otherwise.

Definition at line 968 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

2.37.3.14 `template<typename _Bi_iter > bool std::operator< (const
sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter
>::value_type const & __rhs) [inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

lhs A regular expression submatch.

rhs A const string reference.

Returns

true if *lhs* precedes *rhs*, false otherwise.

Definition at line 1354 of file regex.h.

2.37.3.15 `template<typename _Bi_iter > bool std::operator< (typename
iterator_traits< _Bi_iter >::value_type const * __lhs, const
sub_match< _Bi_iter > & __rhs) [inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

lhs A string.

rhs A regular expression submatch.

Returns

true if *lhs* precedes *rhs*, false otherwise.

Definition at line 1132 of file regex.h.

2.37.3.16 `template<typename _Bi_iter > bool std::operator< (const
sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter
>::value_type const * __rhs) [inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

lhs A regular expression submatch.

rhs A string.

Returns

true if *lhs* precedes *rhs*, false otherwise.

Definition at line 1206 of file regex.h.

2.37.3.17 `template<typename _Ch_type , typename _Ch_traits , typename
_Bi_iter > basic_ostream<_Ch_type, _Ch_traits>& std::operator<<
(basic_ostream<_Ch_type, _Ch_traits > & __os, const
sub_match<_Bi_iter > & __m) [inline]`

Inserts a matched string into an output stream.

Parameters

os The output stream.

m A submatch string.

Returns

the output stream with the submatch string inserted.

Definition at line 1405 of file regex.h.

2.37.3.18 `template<typename _Bi_iter > bool std::operator<= (typename
iterator_traits<_Bi_iter >::value_type const & __lhs, const
sub_match<_Bi_iter > & __rhs) [inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

lhs A string.

rhs A regular expression submatch.

Returns

true if *lhs* does not succeed *rhs*, false otherwise.

Definition at line 1316 of file regex.h.

2.37.3.19 `template<typename _Bi_iter , class _Ch_traits , class _Ch_alloc >
bool std::operator<= (const sub_match< _Bi_iter > & __lhs, const
basic_string< typename iterator_traits< _Bi_iter >::value_type,
_Ch_traits, _Ch_alloc > & __rhs) [inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

lhs A regular expression submatch.

rhs A string.

Returns

true if *lhs* does not succeed *rhs*, false otherwise.

Definition at line 1092 of file regex.h.

2.37.3.20 `template<typename _BiIter > bool std::operator<= (const
sub_match< _BiIter > & __lhs, const sub_match< _BiIter > &
__rhs) [inline]`

Tests the ordering of two regular expression submatches.

Parameters

lhs First regular expression submatch.

rhs Second regular expression submatch.

Returns

true if *lhs* does not succeed *rhs*, false otherwise.

Definition at line 903 of file regex.h.

2.37.3.21 `template<typename _Bi_iter > bool std::operator<= (const
sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter
>::value_type const & __rhs) [inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

lhs A regular expression submatch.

rhs A const string reference.

Returns

true if *lhs* does not succeed *rhs*, false otherwise.

Definition at line 1390 of file regex.h.

2.37.3.22 `template<typename _Bi_iter > bool std::operator<= (typename
iterator_traits< _Bi_iter >::value_type const * __lhs, const
sub_match< _Bi_iter > & __rhs) [inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

lhs A string.

rhs A regular expression submatch.

Returns

true if *lhs* does not succeed *rhs*, false otherwise.

Definition at line 1168 of file regex.h.

2.37.3.23 `template<typename _Bi_iter , typename _Ch_traits , typename
_Ch_alloc > bool std::operator<= (const basic_string< typename
iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &
__lhs, const sub_match< _Bi_iter > & __rhs) [inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

lhs A string.

rhs A regular expression submatch.

Returns

true if *lhs* does not succeed *rhs*, false otherwise.

Definition at line 1007 of file `regex.h`.

References `std::sub_match<_BiIter>::str()`.

2.37.3.24 `template<typename _Bi_iter> bool std::operator<= (const
sub_match<_Bi_iter> & __lhs, typename iterator_traits<_Bi_iter>
>::value_type const * __rhs) [inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

lhs A regular expression submatch.

rhs A string.

Returns

true if *lhs* does not succeed *rhs*, false otherwise.

Definition at line 1242 of file `regex.h`.

2.37.3.25 `template<typename _Bi_iter> bool std::operator== (typename
iterator_traits<_Bi_iter>::value_type const & __lhs, const
sub_match<_Bi_iter> & __rhs) [inline]`

Tests the equivalence of a string and a regular expression submatch.

Parameters

lhs A string.

rhs A regular expression submatch.

Returns

true if *lhs* is equivalent to *rhs*, false otherwise.

Definition at line 1255 of file `regex.h`.

References `std::sub_match<_BiIter>::str()`.

2.37.3.26 `template<typename _Bi_iter , typename _Ch_traits , typename
_Ch_alloc > bool std::operator==(const sub_match< _Bi_iter >
& __lhs, const basic_string< typename iterator_traits< _Bi_iter
>::value_type, _Ch_traits, _Ch_alloc > & __rhs) [inline]`

Tests the equivalence of a regular expression submatch and a string.

Parameters

lhs A regular expression submatch.

rhs A string.

Returns

true if *lhs* is equivalent to *rhs*, false otherwise.

Definition at line 1021 of file regex.h.

References `std::sub_match< _BiIter >::str()`.

2.37.3.27 `template<typename _Bi_iter > bool std::operator==(const
sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter
>::value_type const & __rhs) [inline]`

Tests the equivalence of a regular expression submatch and a string.

Parameters

lhs A regular expression submatch.

rhs A const string reference.

Returns

true if *lhs* is equivalent to *rhs*, false otherwise.

Definition at line 1329 of file regex.h.

References `std::sub_match< _BiIter >::str()`.

2.37.3.28 `template<typename _Bi_iter , typename _Ch_traits , typename
_Ch_alloc > bool std::operator==(const basic_string< typename
iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &
__lhs, const sub_match< _Bi_iter > & __rhs) [inline]`

Tests the equivalence of a string and a regular expression submatch.

Parameters

lhs A string.

rhs A regular expression submatch.

Returns

true if *lhs* is equivalent to *rhs*, false otherwise.

Definition at line 940 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

2.37.3.29 `template<typename _Bi_iter > bool std::operator==(typename
iterator_traits<_Bi_iter>::value_type const * __lhs, const
sub_match<_Bi_iter> & __rhs) [inline]`

Tests the equivalence of a C string and a regular expression submatch.

Parameters

lhs A C string.

rhs A regular expression submatch.

Returns

true if *lhs* is equivalent to *rhs*, false otherwise.

Definition at line 1107 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

2.37.3.30 `template<typename _Bi_iter, typename _Allocator > bool
std::operator==(const match_results<_Bi_iter, _Allocator> &
__m1, const match_results<_Bi_iter, _Allocator> & __m2)
[inline]`

Compares two [match_results](#) for equality.

Returns

true if the two objects refer to the same match, false otherwise.

Todo

Implement this function.

2.37.3.31 `template<typename _BiIter > bool std::operator==(const
sub_match<_BiIter > & __lhs, const sub_match<_BiIter > &
__rhs) [inline]`

Tests the equivalence of two regular expression submatches.

Parameters

lhs First regular expression submatch.

rhs Second regular expression submatch.

Returns

true if *lhs* is equivalent to *rhs*, false otherwise.

Definition at line 867 of file regex.h.

References `std::sub_match<_BiIter >::compare()`.

2.37.3.32 `template<typename _Bi_iter > bool std::operator==(const
sub_match<_Bi_iter > & __lhs, typename iterator_traits<_Bi_iter
>::value_type const * __rhs) [inline]`

Tests the equivalence of a regular expression submatch and a string.

Parameters

lhs A regular expression submatch.

rhs A pointer to a string?

Returns

true if *lhs* is equivalent to *rhs*, false otherwise.

Definition at line 1181 of file regex.h.

References `std::sub_match<_BiIter >::str()`.

2.37.3.33 `template<typename _Bi_iter , class _Ch_traits , class _Ch_alloc >
bool std::operator> (const sub_match< _Bi_iter > & __lhs, const
basic_string< typename iterator_traits< _Bi_iter >::value_type,
_Ch_traits, _Ch_alloc > & __rhs) [inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

lhs A regular expression submatch.

rhs A string.

Returns

true if *lhs* succeeds *rhs*, false otherwise.

Definition at line 1064 of file regex.h.

References `std::sub_match< _BiIter >::str()`.

2.37.3.34 `template<typename _Bi_iter > bool std::operator> (typename
iterator_traits< _Bi_iter >::value_type const & __lhs, const
sub_match< _Bi_iter > & __rhs) [inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

lhs A string.

rhs A regular expression submatch.

Returns

true if *lhs* succeeds *rhs*, false otherwise.

Definition at line 1292 of file regex.h.

References `std::sub_match< _BiIter >::str()`.

2.37.3.35 `template<typename _Bi_iter > bool std::operator> (const
sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter
>::value_type const & __rhs) [inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

lhs A regular expression submatch.

rhs A const string reference.

Returns

true if *lhs* succeeds *rhs*, false otherwise.

Definition at line 1366 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

2.37.3.36 `template<typename _Bi_iter , typename _Ch_traits , typename
_Ch_alloc > bool std::operator> (const basic_string< typename
iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &
__lhs, const sub_match< _Bi_iter > & __rhs) [inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

lhs A string.

rhs A regular expression submatch.

Returns

true if *lhs* succeeds *rhs*, false otherwise.

Definition at line 981 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

2.37.3.37 `template<typename _Bi_iter > bool std::operator> (typename
iterator_traits< _Bi_iter >::value_type const * __lhs, const
sub_match< _Bi_iter > & __rhs) [inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

lhs A string.

rhs A regular expression submatch.

Returns

true if *lhs* succeeds *rhs*, false otherwise.

Definition at line 1144 of file regex.h.

References std::sub_match<_BiIter>::str().

2.37.3.38 `template<typename _BiIter> bool std::operator> (const
sub_match<_BiIter> & __lhs, const sub_match<_BiIter> &
__rhs) [inline]`

Tests the ordering of two regular expression submatches.

Parameters

lhs First regular expression submatch.

rhs Second regular expression submatch.

Returns

true if *lhs* succeeds *rhs*, false otherwise.

Definition at line 927 of file regex.h.

References std::sub_match<_BiIter>::compare().

2.37.3.39 `template<typename _Bi_iter> bool std::operator> (const
sub_match<_Bi_iter> & __lhs, typename iterator_traits<_Bi_iter
>::value_type const * __rhs) [inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

lhs A regular expression submatch.

rhs A string.

Returns

true if *lhs* succeeds *rhs*, false otherwise.

Definition at line 1218 of file regex.h.

References std::sub_match<_BiIter>::str().

2.37.3.40 `template<typename _Bi_iter > bool std::operator>= (typename
iterator_traits< _Bi_iter >::value_type const & __lhs, const
sub_match< _Bi_iter > & __rhs) [inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

lhs A string.

rhs A regular expression submatch.

Returns

true if *lhs* does not precede *rhs*, false otherwise.

Definition at line 1304 of file regex.h.

References `std::sub_match< _BiIter >::str()`.

2.37.3.41 `template<typename _Bi_iter , class _Ch_traits , class _Ch_alloc >
bool std::operator>= (const sub_match< _Bi_iter > & __lhs, const
basic_string< typename iterator_traits< _Bi_iter >::value_type,
_Ch_traits, _Ch_alloc > & __rhs) [inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

lhs A regular expression submatch.

rhs A string.

Returns

true if *lhs* does not precede *rhs*, false otherwise.

Definition at line 1078 of file regex.h.

References `std::sub_match< _BiIter >::str()`.

2.37.3.42 `template<typename _Bi_iter > bool std::operator>= (const
sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter
>::value_type const & __rhs) [inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

lhs A regular expression submatch.

rhs A const string reference.

Returns

true if *lhs* does not precede *rhs*, false otherwise.

Definition at line 1378 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

2.37.3.43 `template<typename _Bi_iter> bool std::operator>= (typename
iterator_traits<_Bi_iter>::value_type const * __lhs, const
sub_match<_Bi_iter> & __rhs) [inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

lhs A string.

rhs A regular expression submatch.

Returns

true if *lhs* does not precede *rhs*, false otherwise.

Definition at line 1156 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

2.37.3.44 `template<typename _Bi_iter, typename _Ch_traits, typename
_Ch_alloc> bool std::operator>= (const basic_string< typename
iterator_traits<_Bi_iter>::value_type, _Ch_traits, _Ch_alloc> &
__lhs, const sub_match<_Bi_iter> & __rhs) [inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

lhs A string.

rhs A regular expression submatch.

Returns

true if *lhs* does not precede *rhs*, false otherwise.

Definition at line 994 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

2.37.3.45 `template<typename _BiIter> bool std::operator>= (const
sub_match<_BiIter> & __lhs, const sub_match<_BiIter> &
__rhs) [inline]`

Tests the ordering of two regular expression submatches.

Parameters

lhs First regular expression submatch.

rhs Second regular expression submatch.

Returns

true if *lhs* does not precede *rhs*, false otherwise.

Definition at line 915 of file regex.h.

References `std::sub_match<_BiIter>::compare()`.

2.37.3.46 `template<typename _Bi_iter> bool std::operator>= (const
sub_match<_Bi_iter> & __lhs, typename iterator_traits<_Bi_iter
>::value_type const * __rhs) [inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

lhs A regular expression submatch.

rhs A string.

Returns

true if *lhs* does not precede *rhs*, false otherwise.

Definition at line 1230 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

```

2.37.3.47 template<typename _Ch_traits , typename _Ch_alloc , typename
        _Allocator , typename _Ch_type , typename _Rx_traits >
        bool std::regex_match ( const basic_string< _Ch_type,
        _Ch_traits, _Ch_alloc > & __s, match_results< typename
        basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator,
        _Allocator > & __m, const basic_regex< _Ch_type, _Rx_traits
        > & __re, regex_constants::match_flag_type __flags =
        regex_constants::match_default ) [inline]

```

Determines if there is a match between the regular expression *e* and a string.

Parameters

- s* The string to match.
- m* The match results.
- re* The regular expression.
- flags* Controls how the regular expression is matched.

Return values

- true* A match exists.
- false* Otherwise.

Exceptions

- an* exception of type [regex_error](#).

Definition at line 1903 of file `regex.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::begin()`, `std::basic_string< _CharT, _Traits, _Alloc >::end()`, and `std::regex_match()`.

```

2.37.3.48 template<typename _Ch_type , typename _Allocator ,
        typename _Rx_traits > bool std::regex_match ( const
        _Ch_type * __s, match_results< const _Ch_type *, _Allocator
        > & __m, const basic_regex< _Ch_type, _Rx_traits
        > & __re, regex_constants::match_flag_type __f =
        regex_constants::match_default ) [inline]

```

Determines if there is a match between the regular expression *e* and a C-style null-terminated string.

Parameters

- s* The C-style null-terminated string to match.
- m* The match results.
- re* The regular expression.
- f* Controls how the regular expression is matched.

Return values

- true* A match exists.
- false* Otherwise.

Exceptions

- an* exception of type `regex_error`.

Definition at line 1879 of file `regex.h`.

References `std::regex_match()`.

```
2.37.3.49 template<typename _Ch_type , class _Rx_traits > bool  
std::regex_match ( const _Ch_type * __s, const basic_regex<  
_Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type  
__f = regex_constants::match_default ) [inline]
```

Indicates if there is a match between the regular expression `e` and a C-style null-terminated string.

Parameters

- s* The C-style null-terminated string to match.
- re* The regular expression.
- f* Controls how the regular expression is matched.

Return values

- true* A match exists.
- false* Otherwise.

Exceptions

- an* exception of type `regex_error`.

Definition at line 1926 of file `regex.h`.

References `std::regex_match()`.

```

2.37.3.50 template<typename _Ch_traits , typename _Str_allocator ,
typename _Ch_type , typename _Rx_traits > bool std::regex_match
( const basic_string< _Ch_type, _Ch_traits, _Str_allocator
> & __s, const basic_regex< _Ch_type, _Rx_traits >
& __re, regex_constants::match_flag_type __flags =
regex_constants::match_default ) [inline]

```

Indicates if there is a match between the regular expression *e* and a string.

Parameters

- s* [IN] The string to match.
- re* [IN] The regular expression.
- flags* [IN] Controls how the regular expression is matched.

Return values

- true* A match exists.
- false* Otherwise.

Exceptions

- an* exception of type [regex_error](#).

Definition at line 1948 of file `regex.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::begin()`, `std::basic_string< _CharT, _Traits, _Alloc >::end()`, and `std::regex_match()`.

```

2.37.3.51 template<typename _Bi_iter , typename _Allocator , typename
_Ch_type , typename _Rx_traits > bool std::regex_match
( _Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter,
_Alocator > & __m, const basic_regex< _Ch_type, _Rx_traits
> & __re, regex_constants::match_flag_type __flags =
regex_constants::match_default )

```

Determines if there is a match between the regular expression *e* and all of the character sequence [first, last).

Parameters

- s* Start of the character sequence to match.
- e* One-past-the-end of the character sequence to match.

m The match results.

re The regular expression.

flags Controls how the regular expression is matched.

Return values

true A match exists.

false Otherwise.

Exceptions

an exception of type [regex_error](#).

Todo

Implement this function.

Definition at line 1823 of file `regex.h`.

Referenced by `std::regex_match()`.

```
2.37.3.52  template<typename _Bi_iter , typename _Ch_type , typename
            _Rx_traits > bool std::regex_match ( _Bi_iter __first,
            _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits
            > & __re, regex_constants::match_flag_type __flags =
            regex_constants::match_default )
```

Indicates if there is a match between the regular expression *e* and all of the character sequence [*first*, *last*).

Parameters

first Beginning of the character sequence to match.

last One-past-the-end of the character sequence to match.

re The regular expression.

flags Controls how the regular expression is matched.

Return values

true A match exists.

false Otherwise.

Exceptions

an exception of type [regex_error](#).

Definition at line 1854 of file `regex.h`.

References `std::regex_match()`.

```
2.37.3.53  template<typename _Rx_traits , typename _Ch_type
> basic_string<_Ch_type> std::regex_replace ( const
basic_string<_Ch_type> & __s, const basic_regex<
_Ch_type, _Rx_traits> & __e, const basic_string<_Ch_type
> & __fmt, regex_constants::match_flag_type __flags =
regex_constants::match_default ) [inline]
```

Todo

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Parameters

s

e

fmt

flags

Returns

a copy of string *s* with replacements.

Exceptions

an exception of type `regex_error`.

Definition at line 2127 of file `regex.h`.

References `std::back_inserter()`, `std::basic_string<_CharT, _Traits, _Alloc>::begin()`, `std::basic_string<_CharT, _Traits, _Alloc>::end()`, and `std::regex_replace()`.

```
2.37.3.54  template<typename _Out_iter , typename _Bi_iter , typename
_Rx_traits , typename _Ch_type> _Out_iter std::regex_replace
( _Out_iter __out, _Bi_iter __first, _Bi_iter __last, const
basic_regex<_Ch_type, _Rx_traits> & __e, const basic_string<
_Ch_type> & __fmt, regex_constants::match_flag_type __flags =
regex_constants::match_default ) [inline]
```


Todo

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Parameters

out

first

last

e

fmt

flags

Returns

out

Exceptions

an exception of type `regex_error`.

Todo

Implement this function.

Definition at line 2107 of file `regex.h`.

Referenced by `std::regex_replace()`.

```
2.37.3.55 template<typename _Ch_traits , typename _Ch_alloc , typename
    _Allocator , typename _Ch_type , typename _Rx_traits >
bool std::regex_search ( const basic_string< _Ch_type,
    _Ch_traits, _Ch_alloc > & __s, match_results< typename
    basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator,
    _Allocator > & __m, const basic_regex< _Ch_type,
    _Rx_traits > & __e, regex_constants::match_flag_type __f =
    regex_constants::match_default ) [inline]
```

Searches for a regular expression within a string.

Parameters

s [IN] A C++ string to search for the regex.

m [OUT] The set of regex matches.

e [IN] The regex to search for in *s*.

f [IN] The search flags.

Return values

true A match was found within the string.

false No match was found within the string, the content of *m* is undefined.

Exceptions

an exception of type [regex_error](#).

Definition at line 2081 of file `regex.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::begin()`, `std::basic_string<_CharT, _Traits, _Alloc>::end()`, and `std::regex_search()`.

```
2.37.3.56 template<typename _Bi_iter, typename _Allocator, typename
    _Ch_type, typename _Rx_traits> bool std::regex_search
    ( _Bi_iter __first, _Bi_iter __last, match_results<_Bi_iter,
    _Allocator> & __m, const basic_regex<_Ch_type, _Rx_traits
    > & __re, regex_constants::match_flag_type __flags =
    regex_constants::match_default ) [inline]
```

Searches for a regular expression within a range.

Parameters

first [IN] The start of the string to search.

last [IN] One-past-the-end of the string to search.

m [OUT] The match results.

re [IN] The regular expression to search for.

flags [IN] Search policy flags.

Return values

true A match was found within the string.

false No match was found within the string, the content of *m* is undefined.

Exceptions

an exception of type [regex_error](#).

Todo

Implement this function.

Definition at line 1973 of file regex.h.

Referenced by std::regex_search().

```
2.37.3.57  template<typename _Ch_type , typename _Rx_traits > bool
            std::regex_search ( const _Ch_type * __s, const basic_regex<
            _Ch_type, _Rx_traits > & __e, regex_constants::match_flag_type
            __f= regex_constants::match_default ) [inline]
```

Searches for a regular expression within a C-string.

Parameters

- s* [IN] The C-string to search.
- e* [IN] The regular expression to search for.
- f* [IN] Search policy flags.

Return values

- true* A match was found within the string.
- false* No match was found within the string.

Todo

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Exceptions

- an* exception of type [regex_error](#).

Definition at line 2038 of file regex.h.

References std::regex_search().

```
2.37.3.58  template<typename _Ch_type , class _Allocator , class _Rx_traits >
            bool std::regex_search ( const _Ch_type * __s, match_results< const
            _Ch_type *, _Allocator > & __m, const basic_regex< _Ch_type,
            _Rx_traits > & __e, regex_constants::match_flag_type __f=
            regex_constants::match_default ) [inline]
```

Searches for a regular expression within a C-string.

Parameters

s [IN] A C-string to search for the regex.

m [OUT] The set of regex matches.

e [IN] The regex to search for in *s*.

f [IN] The search flags.

Return values

true A match was found within the string.

false No match was found within the string, the content of *m* is undefined.

Todo

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Exceptions

an exception of type [regex_error](#).

Definition at line 2018 of file `regex.h`.

References `std::regex_search()`.

2.37.3.59 `template<typename _Bi_iter , typename _Ch_type , typename
_Rx_traits > bool std::regex_search (_Bi_iter __first,
_Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits
> & __re, regex_constants::match_flag_type __flags =
regex_constants::match_default) [inline]`

Searches for a regular expression within a range.

Parameters

first [IN] The start of the string to search.

last [IN] One-past-the-end of the string to search.

re [IN] The regular expression to search for.

flags [IN] Search policy flags.

Return values

true A match was found within the string.

false No match was found within the string.

Todo

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Exceptions

an exception of type `regex_error`.

Definition at line 1994 of file `regex.h`.

References `std::regex_search()`.

```
2.37.3.60 template<typename _Ch_traits , typename _String_allocator ,
typename _Ch_type , typename _Rx_traits > bool std::regex_search
( const basic_string< _Ch_type, _Ch_traits, _String_allocator
> & __s, const basic_regex< _Ch_type, _Rx_traits >
& __e, regex_constants::match_flag_type __flags =
regex_constants::match_default ) [inline]
```

Searches for a regular expression within a string.

Parameters

s [IN] The string to search.

e [IN] The regular expression to search for.

flags [IN] Search policy flags.

Return values

true A match was found within the string.

false No match was found within the string.

Todo

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Exceptions

an exception of type `regex_error`.

Definition at line 2058 of file `regex.h`.

References `std::regex_search()`.

2.37.3.61 `template<typename _Ch_type , typename _Rx_traits > void
std::swap (basic_regex< _Ch_type, _Rx_traits > & __lhs,
basic_regex< _Ch_type, _Rx_traits > & __rhs) [inline]`

Swaps the contents of two regular expression objects.

Parameters

lhs First regular expression.

rhs Second regular expression.

Definition at line 737 of file regex.h.

References `std::basic_regex< _Ch_type, _Rx_traits >::swap()`.

2.37.3.62 `template<typename _Bi_iter , typename _Allocator > void std::swap
(match_results< _Bi_iter, _Allocator > & __lhs, match_results<
_Bi_iter, _Allocator > & __rhs) [inline]`

Swaps two match results.

Parameters

lhs A match result.

rhs A match result.

The contents of the two [match_results](#) objects are swapped.

Definition at line 1793 of file regex.h.

References `std::match_results< _Bi_iter, _Allocator >::swap()`.

2.37.3.63 `template<typename _Ch_type > int std::regex_traits< _Ch_type
>::value (_Ch_type __ch, int __radix) const [inherited]`

Converts a digit to an int.

Parameters

ch a character representing a digit.

radix the radix if the numeric conversion (limited to 8, 10, or 16).

Returns

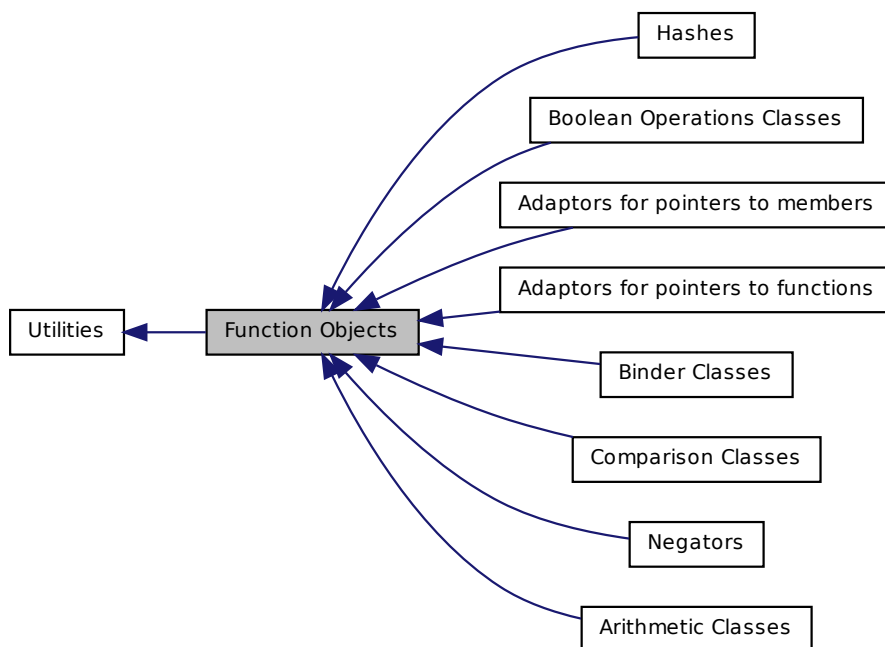
the value represented by the digit `ch` in base `radix` if the character `ch` is a valid digit in base `radix`; otherwise returns `-1`.

Definition at line 320 of file `regex.h`.

References `std::basic_ios<_CharT, _Traits>::fail()`.

2.38 Function Objects

Collaboration diagram for Function Objects:

**Classes**

- `struct std::binary_function<_Arg1, _Arg2, _Result>`
- `class std::function<_Res(_ArgTypes...)>`

*Primary class template for `std::function`.
Polymorphic function wrapper.*

- class `std::reference_wrapper<_Tp>`
Primary class template for `reference_wrapper`.
- struct `std::unary_function<_Arg, _Result>`

Modules

- Binder Classes
- Hashes
- Arithmetic Classes
- Comparison Classes
- Boolean Operations Classes
- Negators
- Adaptors for pointers to functions
- Adaptors for pointers to members

Functions

- template<typename _Tp, typename _Class>
_Mem_fn<_Tp _Class::*> `std::mem_fn` (_Tp _Class::*__pm)

2.38.1 Detailed Description

Function objects, or *functors*, are objects with an `operator()` defined and accessible. They can be passed as arguments to algorithm templates and used in place of a function pointer. Not only is the resulting expressiveness of the library increased, but the generated code can be more efficient than what you might write by hand. When we refer to *functors*, then, generally we include function pointers in the description as well.

Often, functors are only created as temporaries passed to algorithm calls, rather than being created as named variables.

Two examples taken from the standard itself follow. To perform a by-element addition of two vectors `a` and `b` containing `double`, and put the result in `a`, use

```
transform (a.begin(), a.end(), b.begin(), a.begin(), plus<double>());
```

To negate every element in `a`, use

```
transform(a.begin(), a.end(), a.begin(), negate<double>());
```


The addition and negation functions will be inlined directly.

The standard functors are derived from structs named `unary_function` and `binary_function`. These two classes contain nothing but typedefs, to aid in generic (template) programming. If you write your own functors, you might consider doing the same.

2.38.2 Function Documentation

2.38.2.1 `template<typename _Tp , typename _Class > _Mem_fn<_Tp
_Class::*> std::mem_fn (_Tp _Class::* __pm) [inline]`

Returns a function object that forwards to the member pointer *pm*.

Definition at line 812 of file functional.

2.39 Arithmetic Classes

Collaboration diagram for Arithmetic Classes:



Classes

- struct `std::divides<_Tp>`
One of the math functors.
- struct `std::minus<_Tp>`
One of the math functors.
- struct `std::modulus<_Tp>`
One of the math functors.
- struct `std::multiplies<_Tp>`

One of the *math functors*.

- struct `std::negate<_Tp>`

One of the *math functors*.

- struct `std::plus<_Tp>`

One of the *math functors*.

2.39.1 Detailed Description

Because basic math often needs to be done during an algorithm, the library provides functors for those operations. See the documentation for [the base classes](#) for examples of their use.

2.40 Comparison Classes

Collaboration diagram for Comparison Classes:



Classes

- struct `std::equal_to<_Tp>`

One of the *comparison functors*.

- struct `std::greater<_Tp>`

One of the *comparison functors*.

- struct `std::greater_equal<_Tp>`

One of the *comparison functors*.

- struct `std::less<_Tp>`

One of the *comparison functors*.

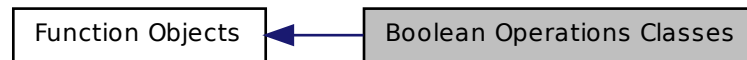
- struct `std::less_equal<_Tp>`
One of the [comparison functors](#).
- struct `std::not_equal_to<_Tp>`
One of the [comparison functors](#).

2.40.1 Detailed Description

The library provides six wrapper functors for all the basic comparisons in C++, like `<`.

2.41 Boolean Operations Classes

Collaboration diagram for Boolean Operations Classes:



Classes

- struct `std::logical_and<_Tp>`
One of the [Boolean operations functors](#).
- struct `std::logical_not<_Tp>`
One of the [Boolean operations functors](#).
- struct `std::logical_or<_Tp>`
One of the [Boolean operations functors](#).

2.41.1 Detailed Description

Here are wrapper functors for Boolean operations: `&&`, `||`, and `!`.

2.42 Negators

Collaboration diagram for Negators:



Classes

- class `std::binary_negate<_Predicate>`
One of the [negation functors](#).
- class `std::unary_negate<_Predicate>`
One of the [negation functors](#).

Functions

- `template<typename _Predicate>`
`unary_negate<_Predicate> std::not1 (const _Predicate &__pred)`
- `template<typename _Predicate>`
`binary_negate<_Predicate> std::not2 (const _Predicate &__pred)`

2.42.1 Detailed Description

The functions `not1` and `not2` each take a predicate functor and return an instance of `unary_negate` or `binary_negate`, respectively. These classes are functors whose `operator()` performs the stored predicate function and then returns the negation of the result.

For example, given a vector of integers and a trivial predicate,

```

struct IntGreaterThanThree
: public std::unary_function<int, bool>
{
    bool operator() (int x) { return x > 3; }
};

std::find_if (v.begin(), v.end(), not1(IntGreaterThanThree()));
  
```

The call to `find_if` will locate the first index (i) of `v` for which `!(v[i] > 3)` is true.

The `not1/unary_negate` combination works on predicates taking a single argument. The `not2/binary_negate` combination works on predicates which take two arguments.

2.42.2 Function Documentation

2.42.2.1 `template<typename _Predicate > unary_negate<_Predicate>
std::not1 (const _Predicate & __pred) [inline]`

One of the [negation functors](#).

Definition at line 370 of file `stl_function.h`.

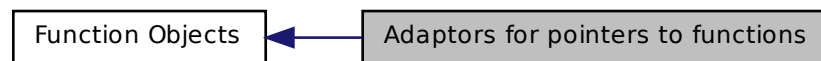
2.42.2.2 `template<typename _Predicate > binary_negate<_Predicate>
std::not2 (const _Predicate & __pred) [inline]`

One of the [negation functors](#).

Definition at line 395 of file `stl_function.h`.

2.43 Adaptors for pointers to functions

Collaboration diagram for Adaptors for pointers to functions:



Classes

- class `std::pointer_to_binary_function< _Arg1, _Arg2, _Result >`
One of the [adaptors for function pointers](#).

- class `std::pointer_to_unary_function<_Arg, _Result>`

One of the [adaptors for function pointers](#).

Functions

- `template<typename _Arg, typename _Result>`
`pointer_to_unary_function<_Arg, _Result> std::ptr_fun (_Result(*)(__x)(-`
`Arg))`
- `template<typename _Arg1, typename _Arg2, typename _Result>`
`pointer_to_binary_function<_Arg1, _Arg2, _Result> std::ptr_fun (-`
`Result(*)(__x)(_Arg1, _Arg2))`

2.43.1 Detailed Description

The advantage of function objects over pointers to functions is that the objects in the standard library declare nested typedefs describing their argument and result types with uniform names (e.g., `result_type` from the base classes [unary_function](#) and [binary_function](#)). Sometimes those typedefs are required, not just optional.

Adaptors are provided to turn pointers to unary (single-argument) and binary (double-argument) functions into function objects. The long-winded functor [pointer_to_unary_function](#) is constructed with a function pointer `f`, and its `operator()` called with argument `x` returns `f(x)`. The functor [pointer_to_binary_function](#) does the same thing, but with a double-argument `f` and `operator()`.

The function `ptr_fun` takes a pointer-to-function `f` and constructs an instance of the appropriate functor.

2.43.2 Function Documentation

2.43.2.1 `template<typename _Arg, typename _Result>`
`pointer_to_unary_function<_Arg, _Result> std::ptr_fun (`
`_Result(*)(_Arg) __x) [inline]`

One of the [adaptors for function pointers](#).

Definition at line 443 of file `stl_function.h`.

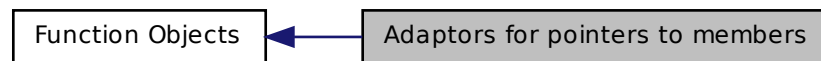
```
2.43.2.2  template<typename _Arg1, typename _Arg2, typename _Result >
           pointer_to_binary_function<_Arg1, _Arg2, _Result> std::ptr_fun (
           _Result(*)(_Arg1, _Arg2) __x )  [inline]
```

One of the [adaptors for function pointers](#).

Definition at line 469 of file stl_function.h.

2.44 Adaptors for pointers to members

Collaboration diagram for Adaptors for pointers to members:



Classes

- class `std::const_mem_fun1_ref_t<_Ret, _Tp, _Arg >`
One of the [adaptors for member /// pointers](#).
- class `std::const_mem_fun1_t<_Ret, _Tp, _Arg >`
One of the [adaptors for member /// pointers](#).
- class `std::const_mem_fun_ref_t<_Ret, _Tp >`
One of the [adaptors for member /// pointers](#).
- class `std::const_mem_fun_t<_Ret, _Tp >`
One of the [adaptors for member /// pointers](#).
- class `std::mem_fun1_ref_t<_Ret, _Tp, _Arg >`
One of the [adaptors for member /// pointers](#).
- class `std::mem_fun1_t<_Ret, _Tp, _Arg >`
One of the [adaptors for member /// pointers](#).

- class `std::mem_fun_ref_t<_Ret, _Tp>`
One of the adaptors for member /// pointers.
- class `std::mem_fun_t<_Ret, _Tp>`
One of the adaptors for member /// pointers.

Functions

- `template<typename _Ret, typename _Tp>`
`mem_fun_t<_Ret, _Tp> std::mem_fun (_Ret(_Tp::*__f)())`
- `template<typename _Ret, typename _Tp, typename _Arg>`
`mem_fun1_t<_Ret, _Tp, _Arg> std::mem_fun (_Ret(_Tp::*__f)(_Arg))`
- `template<typename _Ret, typename _Tp, typename _Arg>`
`mem_fun1_ref_t<_Ret, _Tp, _Arg> std::mem_fun_ref (_Ret(_Tp::*__f)(-Arg))`
- `template<typename _Ret, typename _Tp>`
`mem_fun_ref_t<_Ret, _Tp> std::mem_fun_ref (_Ret(_Tp::*__f)())`

2.44.1 Detailed Description

There are a total of $8 = 2^3$ function objects in this family. (1) Member functions taking no arguments vs member functions taking one argument. (2) Call through pointer vs call through reference. (3) Const vs non-const member function.

All of this complexity is in the function objects themselves. You can ignore it by using the helper function `mem_fun` and `mem_fun_ref`, which create whichever type of adaptor is appropriate.

2.45 Heap

Collaboration diagram for Heap:



Functions

- `template<typename _RandomAccessIterator >`
`bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`
`last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`
`last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`_RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`_RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator`
`__last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator`
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator _-`
`__last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator _-`
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator`
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator`
`__last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator _-`
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator _-`
`__last)`

2.45.1 Function Documentation

2.45.1.1 `template<typename _RandomAccessIterator > bool std::is_heap (`
 `_RandomAccessIterator __first, _RandomAccessIterator __last)`
 `[inline]`

Determines whether a range is a heap.

Parameters

first Start of range.

last End of range.

Returns

True if range is a heap, false otherwise.

Definition at line 560 of file `stl_heap.h`.

References `std::is_heap_until()`.

2.45.1.2 `template<typename _RandomAccessIterator , typename`
 `_Compare > bool std::is_heap (_RandomAccessIterator __first,`
 `_RandomAccessIterator __last, _Compare __comp) [inline]`

Determines whether a range is a heap using comparison functor.

Parameters

first Start of range.

last End of range.

comp Comparison functor to use.

Returns

True if range is a heap, false otherwise.

Definition at line 573 of file `stl_heap.h`.

References `std::is_heap_until()`.

2.45.1.3 `template<typename _RandomAccessIterator >
_RandomAccessIterator std::is_heap_until (_RandomAccessIterator
__first, _RandomAccessIterator __last) [inline]`

Search the end of a heap.

Parameters

first Start of range.

last End of range.

Returns

An iterator pointing to the first element not in the heap.

This operation returns the last iterator *i* in [*first*, *last*) for which the range [*first*, *i*) is a heap.

Definition at line 512 of file `stl_heap.h`.

References `std::distance()`.

2.45.1.4 `template<typename _RandomAccessIterator , typename _Compare >
_RandomAccessIterator std::is_heap_until (_RandomAccessIterator
__first, _RandomAccessIterator __last, _Compare __comp)
[inline]`

Search the end of a heap using comparison functor.

Parameters

first Start of range.

last End of range.

comp Comparison functor to use.

Returns

An iterator pointing to the first element not in the heap.

This operation returns the last iterator *i* in [*first*, *last*) for which the range [*first*, *i*) is a heap. Comparisons are made using *comp*.

Definition at line 538 of file `stl_heap.h`.

References `std::distance()`.

Referenced by `std::is_heap()`.

2.45.1.5 `template<typename _RandomAccessIterator > void std::make_heap (`
`_RandomAccessIterator __first, _RandomAccessIterator __last)`

Construct a heap over a range.

Parameters

first Start of heap.

last End of heap.

This operation makes the elements in [first,last) into a heap.

Definition at line 375 of file stl_heap.h.

2.45.1.6 `template<typename _RandomAccessIterator , typename _Compare`
`> void std::make_heap (_RandomAccessIterator __first,`
`_RandomAccessIterator __last, _Compare __comp)`

Construct a heap over a range using comparison functor.

Parameters

first Start of heap.

last End of heap.

comp Comparison functor to use.

This operation makes the elements in [first,last) into a heap. Comparisons are made using comp.

Definition at line 415 of file stl_heap.h.

Referenced by std::__heap_select(), std::partial_sort_copy(), and std::priority_queue<_Tp, _Sequence, _Compare >::priority_queue().

2.45.1.7 `template<typename _RandomAccessIterator > void std::pop_heap (`
`_RandomAccessIterator __first, _RandomAccessIterator __last)`
`[inline]`

Pop an element off a heap.

Parameters

first Start of heap.

last End of heap.

This operation pops the top of the heap. The elements first and last-1 are swapped and [first,last-1) is made into a heap.

Definition at line 278 of file stl_heap.h.

```
2.45.1.8 template<typename _RandomAccessIterator, typename _Compare  
            > void std::pop_heap ( _RandomAccessIterator __first,  
                                _RandomAccessIterator __last, _Compare __comp ) [inline]
```

Pop an element off a heap using comparison functor.

Parameters

first Start of heap.

last End of heap.

comp Comparison functor to use.

This operation pops the top of the heap. The elements first and last-1 are swapped and [first,last-1) is made into a heap. Comparisons are made using comp.

Definition at line 352 of file stl_heap.h.

Referenced by std::priority_queue< _Tp, _Sequence, _Compare >::pop().

```
2.45.1.9 template<typename _RandomAccessIterator, typename _Compare  
            > void std::push_heap ( _RandomAccessIterator __first,  
                                _RandomAccessIterator __last, _Compare __comp ) [inline]
```

Push an element onto a heap using comparison functor.

Parameters

first Start of heap.

last End of heap + element.

comp Comparison functor.

This operation pushes the element at last-1 onto the valid heap over the range [first,last-1). After completion, [first,last) is a valid heap. Compare operations are performed using comp.

Definition at line 205 of file stl_heap.h.

Referenced by std::priority_queue< _Tp, _Sequence, _Compare >::push().

2.45.1.10 `template<typename _RandomAccessIterator > void std::push_heap (`
 `_RandomAccessIterator __first, _RandomAccessIterator __last)`
 `[inline]`

Push an element onto a heap.

Parameters

first Start of heap.

last End of heap + element.

This operation pushes the element at last-1 onto the valid heap over the range [first,last-1). After completion, [first,last) is a valid heap.

Definition at line 156 of file stl_heap.h.

2.45.1.11 `template<typename _RandomAccessIterator , typename _Compare`
 `> void std::sort_heap (_RandomAccessIterator __first,`
 `_RandomAccessIterator __last, _Compare __comp)`

Sort a heap using comparison functor.

Parameters

first Start of heap.

last End of heap.

comp Comparison functor to use.

This operation sorts the valid heap in the range [first,last). Comparisons are made using comp.

Definition at line 483 of file stl_heap.h.

Referenced by std::partial_sort(), and std::partial_sort_copy().

2.45.1.12 `template<typename _RandomAccessIterator > void std::sort_heap (`
 `_RandomAccessIterator __first, _RandomAccessIterator __last)`

Sort a heap.

Parameters

first Start of heap.

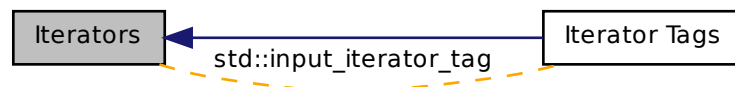
last End of heap.

This operation sorts the valid heap in the range [first,last).

Definition at line 454 of file stl_heap.h.

2.46 Iterators

Collaboration diagram for Iterators:



Classes

- class `std::__has_iterator_category_helper< _Tp >`
Traits class for iterators.
- class `std::back_insert_iterator< _Container >`
Turns assignment into insertion.
- class `std::front_insert_iterator< _Container >`
Turns assignment into insertion.
- struct `std::input_iterator_tag`
Marking input iterators.
- class `std::insert_iterator< _Container >`
Turns assignment into insertion.
- class `std::istream_iterator< _Tp, _CharT, _Traits, _Dist >`
Provides input iterator semantics for streams.
- class `std::istreambuf_iterator< _CharT, _Traits >`
Provides input iterator semantics for streambufs.

- struct `std::iterator<_Category, _Tp, _Distance, _Pointer, _Reference >`
Common iterator class.
- struct `std::iterator_traits<_Tp * >`
Partial specialization for pointer types.
- struct `std::iterator_traits<const _Tp * >`
Partial specialization for const pointer types.
- class `std::move_iterator<_Iterator >`
- class `std::ostream_iterator<_Tp, _CharT, _Traits >`
Provides output iterator semantics for streams.
- class `std::ostreambuf_iterator<_CharT, _Traits >`
Provides output iterator semantics for streambufs.
- class `std::reverse_iterator<_Iterator >`

Modules

- [Iterator Tags](#)

Functions

- template<bool _IsMove, typename _CharT >
__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type **std::__copy_move_a2** (_CharT *__first, _CharT *__last, ostreambuf_iterator< _CharT > __result)
- template<bool _IsMove, typename _CharT >
__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type **std::__copy_move_a2** (const _CharT *__first, const _CharT *__last, ostreambuf_iterator< _CharT > __result)
- template<bool _IsMove, typename _CharT >
__gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type **std::__copy_move_a2** (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, _CharT *__result)
- template<typename _Iter >
iterator_traits< _Iter >::iterator_category **std::__iterator_category** (const _Iter &)
- template<typename _Container >
back_insert_iterator< _Container > **std::back_inserter** (_Container &__x)

- `template<typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_`
`iterator< _CharT > >::__type std::copy (istreambuf_iterator< _CharT > _`
`_first, istreambuf_iterator< _CharT > __last, ostreambuf_iterator< _CharT >`
`__result)`
- `template<typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, istreambuf_`
`iterator< _CharT > >::__type std::find (istreambuf_iterator< _CharT >`
`__first, istreambuf_iterator< _CharT > __last, const _CharT &__val)`
- `template<typename _Container >`
`front_insert_iterator< _Container > std::front_inserter (_Container &__x)`
- `template<typename _Container, typename _Iterator >`
`insert_iterator< _Container > std::inserter (_Container &__x, _Iterator __i)`
- `template<typename _Iterator >`
`move_iterator< _Iterator > std::make_move_iterator (const _Iterator &__i)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator!= (const reverse_iterator< _IteratorL > &__x, const`
`reverse_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`bool std::operator!= (const reverse_iterator< _Iterator > &__x, const reverse_`
`iterator< _Iterator > &__y)`
- `template<class _Tp, class _CharT, class _Traits, class _Dist >`
`bool std::operator!= (const istream_iterator< _Tp, _CharT, _Traits, _Dist > &_`
`__x, const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__y)`
- `template<typename _CharT, typename _Traits >`
`bool std::operator!= (const istreambuf_iterator< _CharT, _Traits > &__a,`
`const istreambuf_iterator< _CharT, _Traits > &__b)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator!= (const move_iterator< _IteratorL > &__x, const move_`
`iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`bool std::operator!= (const move_iterator< _Iterator > &__x, const move_`
`iterator< _Iterator > &__y)`
- `template<typename _Iterator >`
`move_iterator< _Iterator > std::operator+ (typename move_iterator< _Iterator`
`>::difference_type __n, const move_iterator< _Iterator > &__x)`
- `template<typename _Iterator >`
`reverse_iterator< _Iterator > std::operator+ (typename reverse_iterator< _`
`Iterator >::difference_type __n, const reverse_iterator< _Iterator > &__x)`
- `template<typename _Iterator >`
`auto std::operator- (const move_iterator< _Iterator > &__x, const move_`
`iterator< _Iterator > &__y)-> decltype(__x.base()-__y.base())`
- `template<typename _Iterator >`
`reverse_iterator< _Iterator >::difference_type std::operator- (const reverse_`
`iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`

- `template<typename _IteratorL, typename _IteratorR >`
`auto std::operator- (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)-> decltype(__y.base()-__x.base())`
- `template<typename _IteratorL, typename _IteratorR >`
`auto std::operator- (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)-> decltype(__x.base()-__y.base())`
- `template<typename _Iterator >`
`bool std::operator< (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator< (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator< (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`bool std::operator< (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Iterator >`
`bool std::operator<= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator<= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`bool std::operator<= (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator<= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator== (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator== (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _CharT, typename _Traits >`
`bool std::operator== (const istreambuf_iterator< _CharT, _Traits > &__a, const istreambuf_iterator< _CharT, _Traits > &__b)`
- `template<typename _Iterator >`
`bool std::operator== (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Iterator >`
`bool std::operator== (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`

- `template<typename _Tp, typename _CharT, typename _Traits, typename _Dist >`
`bool std::operator== (const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__x, const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator> (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`bool std::operator> (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _Iterator >`
`bool std::operator> (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator> (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`bool std::operator>= (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator>= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`bool std::operator>= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator>= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`

2.46.1 Detailed Description

Abstractions for uniform iterating through various underlying types.

2.46.2 Function Documentation

2.46.2.1 `template<typename _Iter > iterator_traits<_Iter>::iterator_category` `std::__iterator_category (const _Iter &) [inline]`

This function is not a part of the C++ standard but is syntactic sugar for internal library use only.

Definition at line 202 of file `stl_iterator_base_types.h`.

Referenced by `std::advance()`, `std::copy_n()`, `__gnu_cxx::copy_n()`, `std::distance()`, `__gnu_cxx::distance()`, `std::find()`, `std::find_end()`, `std::find_if()`, `std::find_if_not()`,

`std::partition()`, `std::reverse()`, `std::search_n()`, `std::uninitialized_copy_n()`, `__gnu_cxx::uninitialized_copy_n()`, and `std::unique_copy()`.

2.46.2.2 `template<typename _Container > back_insert_iterator<_Container>
std::back_inserter (_Container & __x) [inline]`

Parameters

x A container of arbitrary type.

Returns

An instance of `back_insert_iterator` working on *x*.

This wrapper function helps in creating `back_insert_iterator` instances. Typing the name of the iterator requires knowing the precise full type of the container, which can be tedious and impedes generic programming. Using this function lets you take advantage of automatic template parameter deduction, making the compiler match the correct types for you.

Definition at line 473 of file `stl_iterator.h`.

Referenced by `std::regex_replace()`.

2.46.2.3 `template<typename _Container > front_insert_iterator<_Container>
std::front_inserter (_Container & __x) [inline]`

Parameters

x A container of arbitrary type.

Returns

An instance of `front_insert_iterator` working on *x*.

This wrapper function helps in creating `front_insert_iterator` instances. Typing the name of the iterator requires knowing the precise full type of the container, which can be tedious and impedes generic programming. Using this function lets you take advantage of automatic template parameter deduction, making the compiler match the correct types for you.

Definition at line 563 of file `stl_iterator.h`.

```
2.46.2.4 template<typename _Container , typename _Iterator >
insert_iterator<_Container> std::inserter ( _Container & __x,
    _Iterator __i ) [inline]
```

Parameters

x A container of arbitrary type.

Returns

An instance of [insert_iterator](#) working on *x*.

This wrapper function helps in creating [insert_iterator](#) instances. Typing the name of the iterator requires knowing the precise full type of the container, which can be tedious and impedes generic programming. Using this function lets you take advantage of automatic template parameter deduction, making the compiler match the correct types for you.

Definition at line 677 of file `stl_iterator.h`.

```
2.46.2.5 template<class _Tp , class _CharT , class _Traits , class _Dist > bool
std::operator!=( const istream_iterator< _Tp, _CharT, _Traits, _Dist
    > & __x, const istream_iterator< _Tp, _CharT, _Traits, _Dist > &
    __y ) [inline]
```

Return false if *x* and *y* are both end or not end, or *x* and *y* are the same.

Definition at line 137 of file `stream_iterator.h`.

```
2.46.2.6 template<typename _Iterator > bool std::operator==( const
reverse_iterator< _Iterator > & __x, const reverse_iterator<
    _Iterator > & __y ) [inline]
```

Parameters

x A `reverse_iterator`.

y A `reverse_iterator`.

Returns

A simple bool.

Reverse iterators forward many operations to their underlying base() iterators. Others are implemented in terms of one another.

Definition at line 285 of file `stl_iterator.h`.

References `std::reverse_iterator< _Iterator >::base()`.

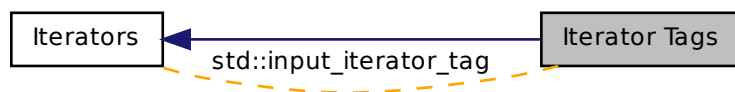
2.46.2.7 `template<typename _Tp, typename _CharT, typename _Traits, typename _Dist > bool std::operator==(const istream_iterator< _Tp, _CharT, _Traits, _Dist > & __x, const istream_iterator< _Tp, _CharT, _Traits, _Dist > & __y) [inline]`

Return true if x and y are both end or not end, or x and y are the same.

Definition at line 130 of file `stream_iterator.h`.

2.47 Iterator Tags

Collaboration diagram for Iterator Tags:



Classes

- struct `std::bidirectional_iterator_tag`
Bidirectional iterators support a superset of forward iterator /// operations.
- struct `std::forward_iterator_tag`
Forward iterators support a superset of input iterator operations.
- struct `std::input_iterator_tag`
Marking input iterators.
- struct `std::output_iterator_tag`

Marking output iterators.

- struct `std::random_access_iterator_tag`

Random-access iterators support a superset of bidirectional /// iterator operations.

2.47.1 Detailed Description

These are empty types, used to distinguish different iterators. The distinction is not made by what they contain, but simply by what they are. Different underlying algorithms can then be used based on the different operations supported by different iterator types.

2.48 Strings

Collaboration diagram for Strings:



Classes

- class `std::basic_string< _CharT, _Traits, _Alloc >`

Managing sequences of characters and character-like objects.

Typedefs

- typedef `basic_string< char >` **`std::string`**
- typedef `basic_string< char16_t >` **`std::u16string`**
- typedef `basic_string< char32_t >` **`std::u32string`**
- typedef `basic_string< wchar_t >` **`std::wstring`**

2.48.1 Typedef Documentation

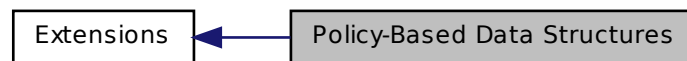
2.48.1.1 typedef basic_string<char32_t> std::u32string

A string of `char16_t`.

Definition at line 80 of file `stringfwd.h`.

2.49 Policy-Based Data Structures

Collaboration diagram for Policy-Based Data Structures:



Classes

- class `__gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, Allocator >`
An abstract basic hash-based associative container.
- class `__gnu_pbds::basic_tree< Key, Mapped, Tag, Node_Update, Policy_Tl, Allocator >`
An abstract basic tree-like (tree, trie) associative container.
- class `__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, Allocator >`
A concrete collision-chaining hash-based associative container.
- class `__gnu_pbds::container_base< Key, Mapped, Tag, Policy_Tl, Allocator >`
An abstract basic associative container.
- class `__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, Allocator >`

A concrete general-probing hash-based associative container.

- `class __gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, Allocator >`

A list-update based associative container.

- `class __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, Allocator >`

A concrete basic tree-based associative container.

- `class __gnu_pbds::trie< Key, Mapped, E_Access_Traits, Tag, Node_Update, Allocator >`

A concrete basic trie-based associative container.

Defines

- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_TREE_NODE_AND_IT_TRAITS_C_DEC`
- `#define PB_DS_TRIE_NODE_AND_ITS_TRAITS`

2.49.1 Detailed Description

This is a library of policy-based elementary data structures: associative containers and priority queues. It is designed for high-performance, flexibility, semantic safety, and conformance to the corresponding containers in std (except for some points where it differs by design).

For details, see: http://gcc.gnu.org/onlinedocs/libstdc++/ext/pb_ds/index.html

2.50 Metaprogramming

Classes

- `struct std::__declval_protector< _Tp >`

declval

- struct `std::__is_member_pointer_helper< _Tp >`
is_member_pointer
- struct `std::add_const< _Tp >`
add_const
- struct `std::add_cv< _Tp >`
add_cv
- struct `std::add_lvalue_reference< _Tp >`
add_lvalue_reference
- struct `std::add_pointer< _Tp >`
add_pointer
- struct `std::add_rvalue_reference< _Tp >`
add_rvalue_reference
- struct `std::add_volatile< _Tp >`
add_volatile
- struct `std::aligned_storage< _Len, _Align >`
Alignment type.
- struct `std::alignment_of< _Tp >`
alignment_of
- struct `std::conditional< _Cond, _Iftrue, _Iffalse >`
conditional
- class `std::decay< _Tp >`
decay
- struct `std::enable_if< bool, _Tp >`
enable_if
- struct `std::extent< typename, _Uint >`
extent
- struct `std::has_nothrow_copy_assign< _Tp >`

has_nothrow_copy_assign

- struct `std::has_nothrow_copy_constructor< _Tp >`
has_nothrow_copy_constructor
- struct `std::has_nothrow_default_constructor< _Tp >`
has_nothrow_default_constructor
- struct `std::has_trivial_copy_assign< _Tp >`
has_trivial_copy_assign
- struct `std::has_trivial_copy_constructor< _Tp >`
has_trivial_copy_constructor
- struct `std::has_trivial_default_constructor< _Tp >`
has_trivial_default_constructor
- struct `std::has_trivial_destructor< _Tp >`
has_trivial_destructor
- struct `std::has_virtual_destructor< _Tp >`
has_virtual_destructor
- struct `std::integral_constant< _Tp, __v >`
integral_constant
- struct `std::is_abstract< _Tp >`
is_abstract
- struct `std::is_arithmetic< _Tp >`
is_arithmetic
- struct `std::is_array< typename >`
is_array
- struct `std::is_base_of< _Base, _Derived >`
is_base_of
- struct `std::is_class< _Tp >`
is_class
- struct `std::is_compound< _Tp >`

is_compound

- struct `std::is_const< typename >`
is_const
- struct `std::is_constructible< _Tp, _Args >`
is_constructible
- struct `std::is_convertible< _From, _To >`
is_convertible
- struct `std::is_empty< _Tp >`
is_empty
- struct `std::is_enum< _Tp >`
is_enum
- struct `std::is_explicitly_convertible< _From, _To >`
is_explicitly_convertible
- struct `std::is_floating_point< _Tp >`
is_floating_point
- struct `std::is_function< typename >`
is_function
- struct `std::is_fundamental< _Tp >`
is_fundamental
- struct `std::is_integral< _Tp >`
is_integral
- struct `std::is_literal_type< _Tp >`
is_literal_type
- struct `std::is_lvalue_reference< typename >`
is_lvalue_reference
- struct `std::is_member_function_pointer< _Tp >`
is_member_function_pointer
- struct `std::is_member_object_pointer< _Tp >`

is_member_object_pointer

- struct `std::is_nothrow_constructible< _Tp, _Args >`
is_nothrow_constructible
- struct `std::is_object< _Tp >`
is_object
- struct `std::is_pod< _Tp >`
is_pod
- struct `std::is_pointer< _Tp >`
is_pointer
- struct `std::is_polymorphic< _Tp >`
is_polymorphic
- struct `std::is_reference< _Tp >`
is_reference
- struct `std::is_rvalue_reference< typename >`
is_rvalue_reference
- struct `std::is_same< typename, typename >`
is_same
- struct `std::is_scalar< _Tp >`
is_scalar
- struct `std::is_signed< _Tp >`
is_signed
- struct `std::is_standard_layout< _Tp >`
is_standard_layout
- struct `std::is_trivial< _Tp >`
is_trivial
- struct `std::is_union< _Tp >`
is_union
- struct `std::is_unsigned< _Tp >`

is_unsigned

- struct `std::is_void< _Tp >`
is_void
- struct `std::is_volatile< typename >`
is_volatile
- struct `std::make_signed< _Tp >`
make_signed
- struct `std::make_unsigned< _Tp >`
make_unsigned
- struct `std::rank< typename >`
rank
- struct `std::remove_all_extents< _Tp >`
remove_all_extents
- struct `std::remove_const< _Tp >`
remove_const
- struct `std::remove_cv< _Tp >`
remove_cv
- struct `std::remove_extent< _Tp >`
remove_extent
- struct `std::remove_pointer< _Tp >`
remove_pointer
- struct `std::remove_reference< _Tp >`
remove_reference
- struct `std::remove_volatile< _Tp >`
remove_volatile

Defines

- `#define _DEFINE_SPEC(_Order, _Trait, _Type, _Value)`
- `#define _DEFINE_SPEC_0_HELPER`
- `#define _DEFINE_SPEC_1_HELPER`
- `#define _DEFINE_SPEC_2_HELPER`
- `#define _GLIBCXX_HAS_NESTED_TYPE(_NTYPE)`

Typedefs

- `typedef integral_constant< bool, false > std::false_type`
- `typedef integral_constant< bool, true > std::true_type`

Functions

- `template<typename _Tp >
add_rvalue_reference< _Tp >::type std::declval () noexcept`

Variables

- `static constexpr _Tp std::integral_constant::value`

2.50.1 Define Documentation**2.50.1.1 `#define _GLIBCXX_HAS_NESTED_TYPE(_NTYPE)`**

Use SFINAE to determine if the type `_Tp` has a publicly-accessible member type `_NTYPE`.

Definition at line 1155 of file `type_traits`.

2.50.2 Typedef Documentation**2.50.2.1 `typedef integral_constant<bool, false> std::false_type`**

typedef for `false_type`

Definition at line 84 of file `type_traits`.

2.50.2.2 typedef integral_constant<bool, true> std::true_type

typedef for true_type

Definition at line 81 of file type_traits.

2.51 Random Number Generators

Collaboration diagram for Random Number Generators:



Classes

- class [std::discard_block_engine<_RandomNumberEngine, __p, __r>](#)
- class [std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>](#)
- class [std::linear_congruential_engine<_UIntType, __a, __c, __m>](#)
A model of a linear congruential random number generator.
- class [std::random_device](#)
- class [std::shuffle_order_engine<_RandomNumberEngine, __k>](#)
Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits __w.

Typedefs

- typedef minstd_rand0 [std::default_random_engine](#)
- typedef shuffle_order_engine< minstd_rand0, 256 > [std::knuth_b](#)
- typedef linear_congruential_engine< uint_fast32_t, 48271UL, 0UL, 2147483647UL > [std::minstd_rand](#)
- typedef linear_congruential_engine< uint_fast32_t, 16807UL, 0UL, 2147483647UL > [std::minstd_rand0](#)

- `typedef mersenne_twister_engine< uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL > std::mt19937`
- `typedef mersenne_twister_engine< uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000ULL, 37, 0xfff7eee000000000ULL, 43, 6364136223846793005ULL > std::mt19937_64`
- `typedef discard_block_engine< ranlux24_base, 223, 23 > std::ranlux24`
- `typedef subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 > std::ranlux24_base`
- `typedef discard_block_engine< ranlux48_base, 389, 11 > std::ranlux48`
- `typedef subtract_with_carry_engine< uint_fast64_t, 48, 5, 12 > std::ranlux48_base`

Functions

- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
bool std::operator!= (const std::linear_congruential_engine< _UIntType, __a, __c, __m > &__lhs, const std::linear_congruential_engine< _UIntType, __a, __c, __m > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __k>
bool std::operator!= (const std::shuffle_order_engine< _RandomNumberEngine, __k > &__lhs, const std::shuffle_order_engine< _RandomNumberEngine, __k > &__rhs)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
bool std::operator!= (const std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__lhs, const std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType >
bool std::operator!= (const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__lhs, const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r>
bool std::operator!= (const std::discard_block_engine< _RandomNumberEngine, __p, __r > &__lhs, const std::discard_block_engine< _RandomNumberEngine, __p, __r > &__rhs)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r>
bool std::operator!= (const std::subtract_with_carry_engine< _UIntType, __w, __s, __r > &__lhs, const std::subtract_with_carry_engine< _UIntType, __w, __s, __r > &__rhs)`

- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__x)`

2.51.1 Detailed Description

These classes define objects which provide random or pseudorandom numbers, either from a discrete or a continuous interval. The random number generator supplied as a part of this library are all uniform random number generators which provide a sequence of random number uniformly distributed over their range.

A number generator is a function object with an `operator()` that takes zero arguments and returns a number.

A compliant random number generator must satisfy the following requirements.

To be documented.

Table 1: Random Number Generator Requirements

2.51.2 Typedef Documentation

2.51.2.1 `typedef linear_congruential_engine<uint_fast32_t, 48271UL, 0UL, 2147483647UL> std::minstd_rand`

An alternative LCR (Lehmer Generator function).

Definition at line 1450 of file `random.h`.

2.51.2.2 `typedef linear_congruential_engine<uint_fast32_t, 16807UL, 0UL, 2147483647UL> std::minstd_rand0`

The classic Minimum Standard `rand0` of Lewis, Goodman, and Miller.

Definition at line 1444 of file `random.h`.

2.51.2.3 `typedef mersenne_twister_engine< uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL> std::mt19937`

The classic Mersenne Twister.

Reference: M. Matsumoto and T. Nishimura, Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator, ACM Transactions on Modeling and Computer Simulation, Vol. 8, No. 1, January 1998, pp 3-30.

Definition at line 1466 of file random.h.

```
2.51.2.4  typedef mersenne_twister_engine< uint_fast64_t, 64, 312, 156,
        31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL,
        17, 0x71d67ffeda60000ULL, 37, 0xfff7eee000000000ULL, 43,
        6364136223846793005ULL> std::mt19937_64
```

An alternative Mersenne Twister.

Definition at line 1478 of file random.h.

2.51.3 Function Documentation

```
2.51.3.1  template<typename _UIntType , _UIntType __a, _UIntType
        __c, _UIntType __m> bool std::operator!= ( const
        std::linear_congruential_engine< _UIntType, __a, __c, __m > &
        __lhs, const std::linear_congruential_engine< _UIntType, __a, __c,
        __m > & __rhs ) [inline]
```

Compares two linear congruential random number generator objects of the same type for inequality.

Parameters

__lhs A linear congruential random number generator object.

__rhs Another linear congruential random number generator object.

Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 338 of file random.h.

```

2.51.3.2 template<typename _RandomNumberEngine , size_t __k>
bool std::operator!= ( const std::shuffle_order_engine<
    _RandomNumberEngine, __k > & __lhs, const
    std::shuffle_order_engine< _RandomNumberEngine, __k > & __rhs
    ) [inline]

```

Compares two `shuffle_order_engine` random number generator objects of the same type for inequality.

Parameters

__lhs A `shuffle_order_engine` random number generator object.

__rhs Another `shuffle_order_engine` random number generator object.

Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 1433 of file `random.h`.

```

2.51.3.3 template<typename _UIntType , size_t __w, size_t __n, size_t __m,
    size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s,
    _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType
    __f> bool std::operator!= ( const std::mersenne_twister_engine<
    _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l,
    __f > & __lhs, const std::mersenne_twister_engine< _UIntType, __w,
    __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > & __rhs )
    [inline]

```

Compares two `% mersenne_twister_engine` random number generator objects of the same type for inequality.

Parameters

__lhs A `% mersenne_twister_engine` random number generator object.

__rhs Another `% mersenne_twister_engine` random number generator object.

Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 568 of file `random.h`.

```

2.51.3.4 template<typename _RandomNumberEngine , size_t
        __w, typename _UIntType > bool std::operator!= ( const
        std::independent_bits_engine< _RandomNumberEngine, __w,
        _UIntType > & __lhs, const std::independent_bits_engine<
        _RandomNumberEngine, __w, _UIntType > & __rhs ) [inline]

```

Compares two `independent_bits_engine` random number generator objects of the same type for inequality.

Parameters

`__lhs` A `independent_bits_engine` random number generator object.
`__rhs` Another `independent_bits_engine` random number generator object.

Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 1183 of file `random.h`.

```

2.51.3.5 template<typename _RandomNumberEngine , size_t __p, size_t
        __r> bool std::operator!= ( const std::discard_block_engine<
        _RandomNumberEngine, __p, __r > & __lhs, const
        std::discard_block_engine< _RandomNumberEngine, __p, __r > &
        __rhs ) [inline]

```

Compares two `discard_block_engine` random number generator objects of the same type for inequality.

Parameters

`__lhs` A `discard_block_engine` random number generator object.
`__rhs` Another `discard_block_engine` random number generator object.

Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 987 of file `random.h`.

2.51.3.6 `template<typename _UIntType, size_t __w, size_t __s, size_t __r> bool
std::operator!=(const std::subtract_with_carry_engine< _UIntType,
__w, __s, __r > & __lhs, const std::subtract_with_carry_engine<
_UIntType, __w, __s, __r > & __rhs) [inline]`

Compares two % `subtract_with_carry_engine` random number generator objects of the same type for inequality.

Parameters

`__lhs` A % `subtract_with_carry_engine` random number generator object.

`__rhs` Another % `subtract_with_carry_engine` random number generator object.

Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 765 of file `random.h`.

2.51.3.7 `template<typename _RandomNumberEngine, size_t __w,
typename _UIntType, typename _CharT, typename _Traits
> std::basic_ostream<_CharT, _Traits>& std::operator<<
(std::basic_ostream<_CharT, _Traits> & __os, const
std::independent_bits_engine<_RandomNumberEngine, __w,
_UIntType> & __x)`

Inserts the current state of a `independent_bits_engine` random number generator engine `__x` into the output stream `__os`.

Parameters

`__os` An output stream.

`__x` A `independent_bits_engine` random number generator engine.

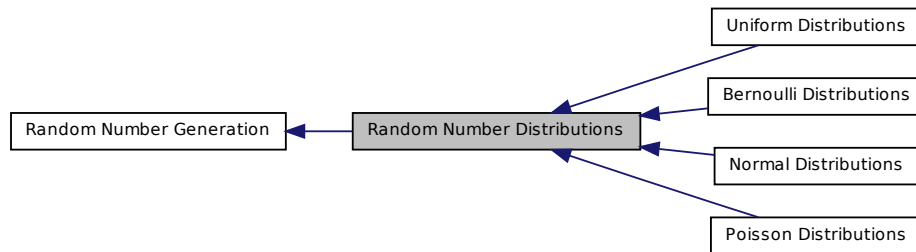
Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 1202 of file `random.h`.

2.52 Random Number Distributions

Collaboration diagram for Random Number Distributions:



Modules

- [Uniform Distributions](#)
- [Normal Distributions](#)
- [Bernoulli Distributions](#)
- [Poisson Distributions](#)

2.53 Uniform Distributions

Collaboration diagram for Uniform Distributions:



Classes

- class [std::uniform_int_distribution< _IntType >](#)

Uniform discrete distribution for random numbers. A discrete random distribution on the range $[min, max]$ with equal probability throughout the range.

- class `std::uniform_real_distribution<_RealType>`

Uniform continuous distribution for random numbers.

Functions

- `template<typename _IntType>`
`bool std::operator!= (const std::uniform_int_distribution<_IntType> &__d1,`
`const std::uniform_int_distribution<_IntType> &__d2)`
- `template<typename _IntType>`
`bool std::operator!= (const std::uniform_real_distribution<_IntType> &__d1,`
`const std::uniform_real_distribution<_IntType> &__d2)`
- `template<typename _RealType, typename _CharT, typename _Traits>`
`std::basic_ostream<_CharT, _Traits> & std::operator<< (std::basic_`
`ostream<_CharT, _Traits> &, const std::uniform_real_distribution<_`
`RealType> &)`
- `template<typename _IntType, typename _CharT, typename _Traits>`
`std::basic_ostream<_CharT, _Traits> & std::operator<< (std::basic_`
`ostream<_CharT, _Traits> &, const std::uniform_int_distribution<_IntType`
`> &)`
- `template<typename _IntType>`
`bool std::operator== (const std::uniform_real_distribution<_IntType> &__d1,`
`const std::uniform_real_distribution<_IntType> &__d2)`
- `template<typename _IntType>`
`bool std::operator== (const std::uniform_int_distribution<_IntType> &__d1,`
`const std::uniform_int_distribution<_IntType> &__d2)`
- `template<typename _RealType, typename _CharT, typename _Traits>`
`std::basic_istream<_CharT, _Traits> & std::operator>> (std::basic_istream<`
`_CharT, _Traits> &, std::uniform_real_distribution<_RealType> &)`
- `template<typename _IntType, typename _CharT, typename _Traits>`
`std::basic_istream<_CharT, _Traits> & std::operator>> (std::basic_istream<`
`_CharT, _Traits> &, std::uniform_int_distribution<_IntType> &)`

2.53.1 Function Documentation

- 2.53.1.1** `template<typename _IntType> bool std::operator!= (const`
`std::uniform_int_distribution<_IntType> & __d1, const`
`std::uniform_int_distribution<_IntType> & __d2) [inline]`

Return true if two uniform integer distributions have different parameters.

Definition at line 1735 of file random.h.

2.53.1.2 `template<typename _IntType > bool std::operator!=(const
std::uniform_real_distribution< _IntType > & __d1, const
std::uniform_real_distribution< _IntType > & __d2) [inline]`

Return true if two uniform real distributions have different parameters.

Definition at line 1916 of file random.h.

2.53.1.3 `template<typename _RealType, typename _CharT, typename _Traits
> std::basic_ostream< _CharT, _Traits > & std::operator<<
(std::basic_ostream< _CharT, _Traits > & __os, const
std::uniform_real_distribution< _RealType > & __x)`

Inserts a `uniform_real_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

`__os` An output stream.

`__x` A `uniform_real_distribution` random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 942 of file random.tcc.

References `std::ios_base::flags()`, `std::left()`, and `std::scientific()`.

2.53.1.4 `template<typename _IntType, typename _CharT, typename _Traits
> std::basic_ostream< _CharT, _Traits > & std::operator<<
(std::basic_ostream< _CharT, _Traits > & __os, const
std::uniform_int_distribution< _IntType > & __x)`

Inserts a `uniform_int_distribution` random number distribution `__x` into the output stream `os`.

Parameters

`__os` An output stream.
`__x` A `uniform_int_distribution` random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 900 of file `random.tcc`.

References `std::left()`, and `std::scientific()`.

2.53.1.5 `template<typename _IntType > bool std::operator==(const
std::uniform_real_distribution< _IntType > & __d1, const
std::uniform_real_distribution< _IntType > & __d2) [inline]`

Return true if two uniform real distributions have the same parameters.

Definition at line 1906 of file `random.h`.

References `std::uniform_real_distribution< _RealType >::param()`.

2.53.1.6 `template<typename _IntType > bool std::operator==(const
std::uniform_int_distribution< _IntType > & __d1, const
std::uniform_int_distribution< _IntType > & __d2) [inline]`

Return true if two uniform integer distributions have the same parameters.

Definition at line 1725 of file `random.h`.

References `std::uniform_int_distribution< _IntType >::param()`.

2.53.1.7 `template<typename _RealType , typename _CharT ,
typename _Traits > std::basic_istream< _CharT, _Traits > &
std::operator>> (std::basic_istream< _CharT, _Traits > & __is,
std::uniform_real_distribution< _RealType > & __x)`

Extracts a `uniform_real_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A `uniform_real_distribution` random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

Definition at line 966 of file `random.tcc`.

References `std::ios_base::flags()`, `std::uniform_real_distribution< _RealType >::param()`, and `std::skipws()`.

2.53.1.8 `template<typename _IntType , typename _CharT , typename _Traits > std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > & __is, std::uniform_int_distribution< _IntType > & __x)`

Extracts a `uniform_int_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A `uniform_int_distribution` random number generator engine.

Returns

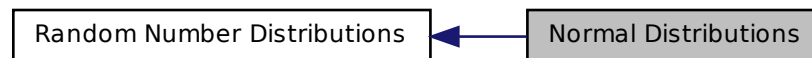
The input stream with `__x` extracted or in an error state.

Definition at line 921 of file `random.tcc`.

References `std::dec()`, `std::ios_base::flags()`, `std::uniform_int_distribution< _IntType >::param()`, and `std::skipws()`.

2.54 Normal Distributions

Collaboration diagram for Normal Distributions:



Classes

- class `std::cauchy_distribution<_RealType>`
A *cauchy_distribution* random number distribution.
- class `std::chi_squared_distribution<_RealType>`
A *chi_squared_distribution* random number distribution.
- class `std::fisher_f_distribution<_RealType>`
A *fisher_f_distribution* random number distribution.
- class `std::gamma_distribution<_RealType>`
A *gamma* continuous distribution for random numbers.
- class `std::lognormal_distribution<_RealType>`
A *lognormal_distribution* random number distribution.
- class `std::normal_distribution<_RealType>`
A *normal* continuous distribution for random numbers.
- class `std::student_t_distribution<_RealType>`
A *student_t_distribution* random number distribution.

Functions

- `template<typename _RealType>`
`bool std::operator!= (const std::normal_distribution<_RealType> &__d1,`
`const std::normal_distribution<_RealType> &__d2)`
- `template<typename _RealType>`
`bool std::operator!= (const std::lognormal_distribution<_RealType> &__d1,`
`const std::lognormal_distribution<_RealType> &__d2)`
- `template<typename _RealType>`
`bool std::operator!= (const std::chi_squared_distribution<_RealType> &__d1,`
`const std::chi_squared_distribution<_RealType> &__d2)`
- `template<typename _RealType>`
`bool std::operator!= (const std::fisher_f_distribution<_RealType> &__d1,`
`const std::fisher_f_distribution<_RealType> &__d2)`
- `template<typename _RealType>`
`bool std::operator!= (const std::student_t_distribution<_RealType> &__d1,`
`const std::student_t_distribution<_RealType> &__d2)`
- `template<typename _RealType>`
`bool std::operator!= (const std::cauchy_distribution<_RealType> &__d1,`
`const std::cauchy_distribution<_RealType> &__d2)`

- `template<typename _RealType >`
`bool std::operator!= (const std::gamma_distribution< _RealType > &__d1,`
`const std::gamma_distribution< _RealType > &__d2)`
- `template<typename _RealType , typename _CharT , typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_`
`ostream< _CharT, _Traits > &, const std::cauchy_distribution< _RealType >`
`&)`
- `template<typename _RealType >`
`bool std::operator== (const std::cauchy_distribution< _RealType > &__d1,`
`const std::cauchy_distribution< _RealType > &__d2)`
- `template<typename _RealType , typename _CharT , typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<`
`_CharT, _Traits > &, std::cauchy_distribution< _RealType > &)`

2.54.1 Function Documentation

2.54.1.1 `template<typename _RealType > bool std::operator!= (const`
`std::normal_distribution< _RealType > & __d1, const`
`std::normal_distribution< _RealType > & __d2) [inline]`

Return true if two normal distributions are different.

Definition at line 2136 of file random.h.

2.54.1.2 `template<typename _RealType > bool std::operator!= (const`
`std::lognormal_distribution< _RealType > & __d1, const`
`std::lognormal_distribution< _RealType > & __d2) [inline]`

Return true if two lognormal distributions are different.

Definition at line 2312 of file random.h.

2.54.1.3 `template<typename _RealType > bool std::operator!= (const`
`std::chi_squared_distribution< _RealType > & __d1, const`
`std::chi_squared_distribution< _RealType > & __d2) [inline]`

Return true if two Chi-squared distributions are different.

Definition at line 2669 of file random.h.

2.54.1.4 `template<typename _RealType > bool std::operator!= (const
std::fisher_f_distribution< _RealType > & __d1, const
std::fisher_f_distribution< _RealType > & __d2) [inline]`

Return true if two Fisher f distributions are diferent.

Definition at line 3024 of file random.h.

2.54.1.5 `template<typename _RealType > bool std::operator!= (const
std::student_t_distribution< _RealType > & __d1, const
std::student_t_distribution< _RealType > & __d2) [inline]`

Return true if two Student t distributions are different.

Definition at line 3198 of file random.h.

2.54.1.6 `template<typename _RealType > bool std::operator!= (const
std::cauchy_distribution< _RealType > & __d1, const
std::cauchy_distribution< _RealType > & __d2) [inline]`

Return true if two Cauchy distributions have different parameters.

Definition at line 2808 of file random.h.

2.54.1.7 `template<typename _RealType > bool std::operator!= (const
std::gamma_distribution< _RealType > & __d1, const
std::gamma_distribution< _RealType > & __d2) [inline]`

Return true if two gamma distributions are different.

Definition at line 2504 of file random.h.

2.54.1.8 `template<typename _RealType , typename _CharT , typename _Traits
> std::basic_ostream< _CharT, _Traits > & std::operator<<
(std::basic_ostream< _CharT, _Traits > & __os, const
std::cauchy_distribution< _RealType > & __x)`

Inserts a `cauchy_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

`__os` An output stream.
`__x` A `cauchy_distribution` random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 1864 of file `random.tcc`.

References `std::left()`, and `std::scientific()`.

2.54.1.9 `template<typename _RealType> bool std::operator==(const std::cauchy_distribution<_RealType> & __d1, const std::cauchy_distribution<_RealType> & __d2) [inline]`

Return true if two Cauchy distributions have the same parameters.

Definition at line 2798 of file `random.h`.

References `std::cauchy_distribution<_RealType>::param()`.

2.54.1.10 `template<typename _RealType, typename _CharT, typename _Traits> std::basic_istream<_CharT, _Traits> & std::operator>>(std::basic_istream<_CharT, _Traits> & __is, std::cauchy_distribution<_RealType> & __x)`

Extracts a `cauchy_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

`__is` An input stream.
`__x` A `cauchy_distribution` random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

Definition at line 1888 of file random.tcc.

References `std::dec()`, `std::ios_base::flags()`, `std::cauchy_distribution< _RealType >::param()`, and `std::skipws()`.

2.55 Bernoulli Distributions

Collaboration diagram for Bernoulli Distributions:



Classes

- class [std::bernoulli_distribution](#)
A Bernoulli random number distribution.
- class [std::binomial_distribution< _IntType >](#)
A discrete binomial random number distribution.
- class [std::geometric_distribution< _IntType >](#)
A discrete geometric random number distribution.
- class [std::negative_binomial_distribution< _IntType >](#)
A [negative_binomial_distribution](#) random number distribution.

Functions

- `bool std::operator!= (const std::bernoulli_distribution &__d1, const std::bernoulli_distribution &__d2)`
- `template<typename _IntType >
bool std::operator!= (const std::binomial_distribution< _IntType > &__d1,
const std::binomial_distribution< _IntType > &__d2)`

- `template<typename _IntType >`
`bool std::operator!= (const std::negative_binomial_distribution< _IntType >`
`&__d1, const std::negative_binomial_distribution< _IntType > &__d2)`
- `template<typename _IntType >`
`bool std::operator!= (const std::geometric_distribution< _IntType > &__d1,`
`const std::geometric_distribution< _IntType > &__d2)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_-`
`ostream< _CharT, _Traits > &, const std::geometric_distribution< _IntType >`
`&)`
- `template<typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_-`
`ostream< _CharT, _Traits > &, const std::bernoulli_distribution &)`
- `bool std::operator== (const std::bernoulli_distribution &__d1, const`
`std::bernoulli_distribution &__d2)`
- `template<typename _IntType >`
`bool std::operator== (const std::geometric_distribution< _IntType > &__d1,`
`const std::geometric_distribution< _IntType > &__d2)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<`
`_CharT, _Traits > &, std::geometric_distribution< _IntType > &)`
- `template<typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<`
`_CharT, _Traits > &__is, std::bernoulli_distribution &__x)`

2.55.1 Function Documentation

2.55.1.1 `bool std::operator!= (const std::bernoulli_distribution & __d1, const
std::bernoulli_distribution & __d2) [inline]`

Return true if two Bernoulli distributions have different parameters.

Definition at line 3346 of file random.h.

2.55.1.2 `template<typename _IntType > bool std::operator!= (const
std::binomial_distribution< _IntType > & __d1, const
std::binomial_distribution< _IntType > & __d2) [inline]`

Return true if two binomial distributions are different.

Definition at line 3583 of file random.h.

2.55.1.3 `template<typename _IntType > bool std::operator!=(const
std::negative_binomial_distribution< _IntType > & __d1, const
std::negative_binomial_distribution< _IntType > & __d2)
[inline]`

Return true if two negative binomial distributions are different.

Definition at line 3929 of file random.h.

2.55.1.4 `template<typename _IntType > bool std::operator!=(const
std::geometric_distribution< _IntType > & __d1, const
std::geometric_distribution< _IntType > & __d2) [inline]`

Return true if two geometric distributions have different parameters.

Definition at line 3725 of file random.h.

2.55.1.5 `template<typename _IntType , typename _CharT , typename _Traits
> std::basic_ostream< _CharT, _Traits > & std::operator<<
(std::basic_ostream< _CharT, _Traits > & __os, const
std::geometric_distribution< _IntType > & __x)`

Inserts a `geometric_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

`__os` An output stream.

`__x` A `geometric_distribution` random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 1037 of file random.tcc.

References `std::left()`, and `std::scientific()`.

2.55.1.6 `template<typename _CharT, typename _Traits >
 std::basic_ostream< _CharT, _Traits > & std::operator<<
 (std::basic_ostream< _CharT, _Traits > & __os, const
 std::bernoulli_distribution & __x)`

Inserts a `bernoulli_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

`__os` An output stream.
`__x` A `bernoulli_distribution` random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 987 of file `random.tcc`.

References `std::ios_base::flags()`, `std::left()`, and `std::scientific()`.

2.55.1.7 `bool std::operator==(const std::bernoulli_distribution & __d1,
 const std::bernoulli_distribution & __d2) [inline]`

Return true if two Bernoulli distributions have the same parameters.

Definition at line 3337 of file `random.h`.

References `std::bernoulli_distribution::param()`.

2.55.1.8 `template<typename _IntType > bool std::operator==(const
 std::geometric_distribution< _IntType > & __d1, const
 std::geometric_distribution< _IntType > & __d2) [inline]`

Return true if two geometric distributions have the same parameters.

Definition at line 3715 of file `random.h`.

References `std::geometric_distribution< _IntType >::param()`.

2.55.1.9 `template<typename _IntType , typename _CharT , typename
_Traits > std::basic_istream< _CharT, _Traits > &
std::operator>> (std::basic_istream< _CharT, _Traits > & __is,
std::geometric_distribution< _IntType > & __x)`

Extracts a `geometric_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A `geometric_distribution` random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

Definition at line 1061 of file `random.tcc`.

References `std::ios_base::flags()`, `std::geometric_distribution< _IntType >::param()`, and `std::skipws()`.

2.55.1.10 `template<typename _CharT , typename _Traits >
std::basic_istream< _CharT, _Traits> & std::operator>>
(std::basic_istream< _CharT, _Traits > & __is,
std::bernoulli_distribution & __x)`

Extracts a `bernoulli_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A `bernoulli_distribution` random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

Definition at line 3376 of file `random.h`.

References `std::bernoulli_distribution::param()`.

2.56 Poisson Distributions

Collaboration diagram for Poisson Distributions:



Classes

- class `std::discrete_distribution< _IntType >`
A *discrete_distribution* random number distribution.
- class `std::exponential_distribution< _RealType >`
An *exponential continuous distribution* for random numbers.
- class `std::extreme_value_distribution< _RealType >`
A *extreme_value_distribution* random number distribution.
- class `std::piecewise_constant_distribution< _RealType >`
A *piecewise_constant_distribution* random number distribution.
- class `std::piecewise_linear_distribution< _RealType >`
A *piecewise_linear_distribution* random number distribution.
- class `std::poisson_distribution< _IntType >`
A *discrete Poisson* random number distribution.
- class `std::weibull_distribution< _RealType >`
A *weibull_distribution* random number distribution.

Functions

- template<typename _IntType >
bool `std::operator!=` (const `std::poisson_distribution< _IntType >` &__d1, const `std::poisson_distribution< _IntType >` &__d2)

- `template<typename _RealType >`
`bool std::operator!= (const std::extreme_value_distribution< _RealType > &_`
`_d1, const std::extreme_value_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::piecewise_constant_distribution< _RealType >`
`&__d1, const std::piecewise_constant_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::piecewise_linear_distribution< _RealType >`
`&__d1, const std::piecewise_linear_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::exponential_distribution< _RealType > &__d1,`
`const std::exponential_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::weibull_distribution< _RealType > &__d1,`
`const std::weibull_distribution< _RealType > &__d2)`
- `template<typename _IntType >`
`bool std::operator!= (const std::discrete_distribution< _IntType > &__d1, const`
`std::discrete_distribution< _IntType > &__d2)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_-`
`ostream< _CharT, _Traits > &, const std::weibull_distribution< _RealType >`
`&)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_-`
`ostream< _CharT, _Traits > &, const std::exponential_distribution< _RealType`
`> &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_-`
`ostream< _CharT, _Traits > &, const std::extreme_value_distribution< _`
`RealType > &)`
- `template<typename _RealType >`
`bool std::operator== (const std::exponential_distribution< _RealType > &__d1,`
`const std::exponential_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator== (const std::piecewise_constant_distribution< _RealType >`
`&__d1, const std::piecewise_constant_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator== (const std::piecewise_linear_distribution< _RealType >`
`&__d1, const std::piecewise_linear_distribution< _RealType > &__d2)`
- `template<typename _IntType >`
`bool std::operator== (const std::discrete_distribution< _IntType > &__d1, const`
`std::discrete_distribution< _IntType > &__d2)`
- `template<typename _RealType >`
`bool std::operator== (const std::extreme_value_distribution< _RealType > &_`
`_d1, const std::extreme_value_distribution< _RealType > &__d2)`

- `template<typename _RealType >`
`bool std::operator== (const std::weibull_distribution< _RealType > &__d1,`
`const std::weibull_distribution< _RealType > &__d2)`
- `template<typename _RealType , typename _CharT , typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<`
`_CharT, _Traits > &, std::exponential_distribution< _RealType > &)`
- `template<typename _RealType , typename _CharT , typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<`
`_CharT, _Traits > &, std::weibull_distribution< _RealType > &)`
- `template<typename _RealType , typename _CharT , typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<`
`_CharT, _Traits > &, std::extreme_value_distribution< _RealType > &)`

2.56.1 Function Documentation

2.56.1.1 `template<typename _IntType > bool std::operator!= (const`
`std::poisson_distribution< _IntType > & __d1, const`
`std::poisson_distribution< _IntType > & __d2) [inline]`

Return true if two Poisson distributions are different.

Definition at line 4117 of file random.h.

2.56.1.2 `template<typename _RealType > bool std::operator!= (const`
`std::extreme_value_distribution< _RealType > & __d1, const`
`std::extreme_value_distribution< _RealType > & __d2) [inline]`

Return true if two extreme value distributions have different parameters.

Definition at line 4617 of file random.h.

2.56.1.3 `template<typename _RealType > bool std::operator!= (const`
`std::piecewise_constant_distribution< _RealType > & __d1, const`
`std::piecewise_constant_distribution< _RealType > & __d2)`
`[inline]`

Return true if two piecewise constant distributions have different parameters.

Definition at line 5088 of file random.h.

2.56.1.4 `template<typename _RealType > bool std::operator!= (const
std::piecewise_linear_distribution< _RealType > & __d1, const
std::piecewise_linear_distribution< _RealType > & __d2)
[inline]`

Return true if two piecewise linear distributions have different parameters.

Definition at line 5330 of file random.h.

2.56.1.5 `template<typename _RealType > bool std::operator!= (const
std::exponential_distribution< _RealType > & __d1, const
std::exponential_distribution< _RealType > & __d2) [inline]`

Return true if two exponential distributions have different parameters.

Definition at line 4267 of file random.h.

2.56.1.6 `template<typename _RealType > bool std::operator!= (const
std::weibull_distribution< _RealType > & __d1, const
std::weibull_distribution< _RealType > & __d2) [inline]`

Return true if two Weibull distributions have different parameters.

Definition at line 4442 of file random.h.

2.56.1.7 `template<typename _IntType > bool std::operator!= (const
std::discrete_distribution< _IntType > & __d1, const
std::discrete_distribution< _IntType > & __d2) [inline]`

Return true if two discrete distributions have different parameters.

Definition at line 4849 of file random.h.

2.56.1.8 `template<typename _RealType , typename _CharT , typename _Traits
> std::basic_ostream< _CharT, _Traits > & std::operator<<
(std::basic_ostream< _CharT, _Traits > & __os, const
std::weibull_distribution< _RealType > & __x)`

Inserts a `weibull_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

`__os` An output stream.
`__x` A `weibull_distribution` random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 2117 of file `random.tcc`.

References `std::left()`, and `std::scientific()`.

2.56.1.9 `template<typename _RealType , typename _CharT , typename _Traits
> std::basic_ostream< _CharT, _Traits > & std::operator<<
(std::basic_ostream< _CharT, _Traits > & __os, const
std::exponential_distribution< _RealType > & __x)`

Inserts a `exponential_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

`__os` An output stream.
`__x` A `exponential_distribution` random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 1597 of file `random.tcc`.

References `std::ios_base::flags()`, `std::left()`, and `std::scientific()`.

2.56.1.10 `template<typename _RealType , typename _CharT , typename
_Traits > std::basic_ostream< _CharT, _Traits > & std::operator<<
(std::basic_ostream< _CharT, _Traits > & __os, const
std::extreme_value_distribution< _RealType > & __x)`

Inserts a `extreme_value_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

- `__os` An output stream.
- `__x` A `extreme_value_distribution` random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 2174 of file `random.tcc`.

References `std::left()`, and `std::scientific()`.

2.56.1.11 `template<typename _RealType > bool std::operator==(const
std::exponential_distribution< _RealType > & __d1, const
std::exponential_distribution< _RealType > & __d2) [inline]`

Return true if two exponential distributions have the same parameters.

Definition at line 4257 of file `random.h`.

References `std::exponential_distribution< _RealType >::param()`.

2.56.1.12 `template<typename _RealType > bool std::operator==(const
std::piecewise_constant_distribution< _RealType > & __d1, const
std::piecewise_constant_distribution< _RealType > & __d2)
[inline]`

Return true if two piecewise constant distributions have the same parameters.

Definition at line 5078 of file `random.h`.

References `std::piecewise_constant_distribution< _RealType >::param()`.

2.56.1.13 `template<typename _RealType > bool std::operator==(const
std::piecewise_linear_distribution< _RealType > & __d1, const
std::piecewise_linear_distribution< _RealType > & __d2)
[inline]`

Return true if two piecewise linear distributions have the same parameters.

Definition at line 5320 of file `random.h`.

References `std::piecewise_linear_distribution< _RealType >::param()`.

2.56.1.14 `template<typename _IntType > bool std::operator==(const
std::discrete_distribution< _IntType > & __d1, const
std::discrete_distribution< _IntType > & __d2) [inline]`

Return true if two discrete distributions have the same parameters.

Definition at line 4839 of file random.h.

References `std::discrete_distribution< _IntType >::param()`.

2.56.1.15 `template<typename _RealType > bool std::operator==(const
std::extreme_value_distribution< _RealType > & __d1, const
std::extreme_value_distribution< _RealType > & __d2)
[inline]`

Return true if two extreme value distributions have the same parameters.

Definition at line 4607 of file random.h.

References `std::extreme_value_distribution< _RealType >::param()`.

2.56.1.16 `template<typename _RealType > bool std::operator==((const
std::weibull_distribution< _RealType > & __d1, const
std::weibull_distribution< _RealType > & __d2) [inline]`

Return true if two Weibull distributions have the same parameters.

Definition at line 4432 of file random.h.

References `std::weibull_distribution< _RealType >::param()`.

2.56.1.17 `template<typename _RealType , typename _CharT ,
typename _Traits > std::basic_istream< _CharT, _Traits > &
std::operator>> (std::basic_istream< _CharT, _Traits > & __is,
std::exponential_distribution< _RealType > & __x)`

Extracts a `exponential_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A exponential_distribution random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

Definition at line 1620 of file random.tcc.

References `std::dec()`, `std::ios_base::flags()`, `std::exponential_distribution<_RealType>::param()`, and `std::skipws()`.

2.56.1.18 `template<typename _RealType , typename _CharT ,
typename _Traits > std::basic_istream< _CharT, _Traits > &
std::operator>> (std::basic_istream< _CharT, _Traits > & __is,
std::weibull_distribution< _RealType > & __x)`

Extracts a weibull_distribution random number distribution `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A weibull_distribution random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

Definition at line 2141 of file random.tcc.

References `std::dec()`, `std::ios_base::flags()`, `std::weibull_distribution<_RealType>::param()`, and `std::skipws()`.

2.56.1.19 `template<typename _RealType , typename _CharT ,
typename _Traits > std::basic_istream< _CharT, _Traits > &
std::operator>> (std::basic_istream< _CharT, _Traits > & __is,
std::extreme_value_distribution< _RealType > & __x)`

Extracts a extreme_value_distribution random number distribution `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A `extreme_value_distribution` random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

Definition at line 2198 of file `random.tcc`.

References `std::dec()`, `std::ios_base::flags()`, `std::extreme_value_distribution< _RealType >::param()`, and `std::skipws()`.

2.57 Random Number Utilities

Collaboration diagram for Random Number Utilities:



Classes

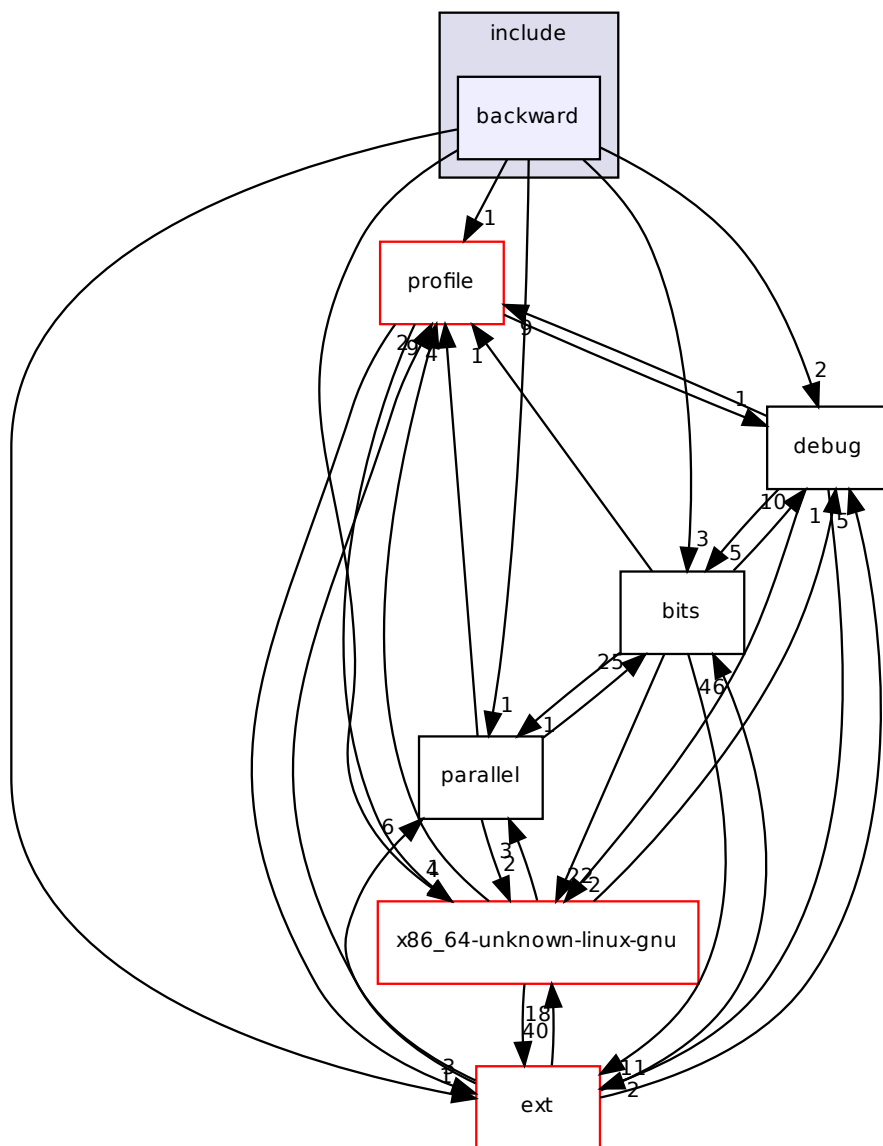
- class [std::seed_seq](#)

The [seed_seq](#) class generates sequences of seeds for random number generators.

3 Directory Documentation

3.1 include/backward/ Directory Reference

Directory dependency graph for include/backward/:

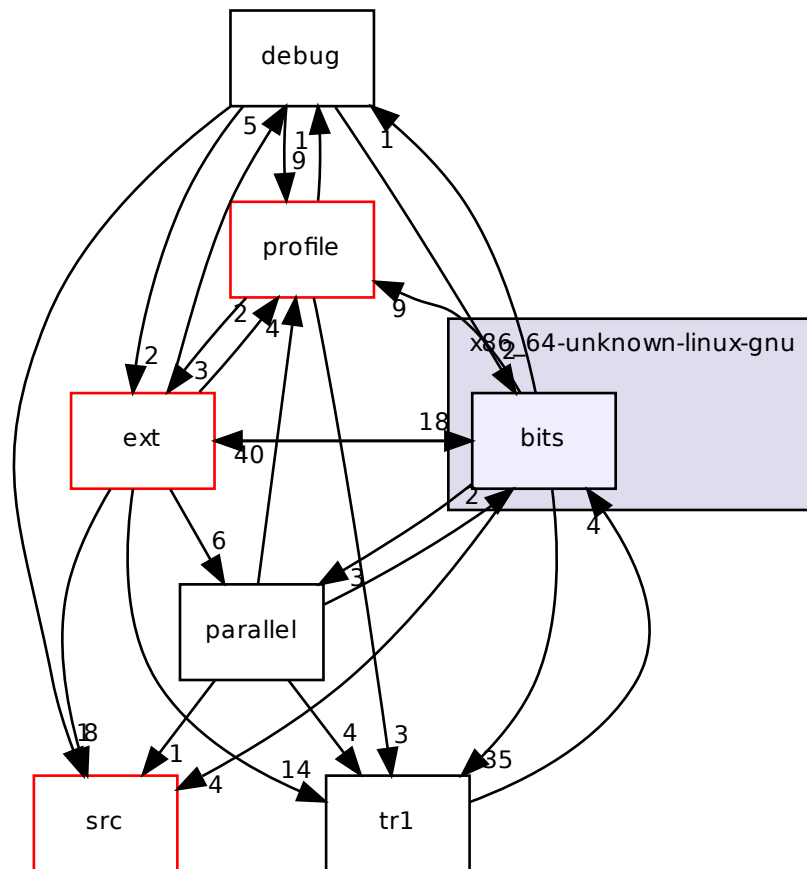


Files

- file [auto_ptr.h](#)
- file [backward_warning.h](#)
- file [binders.h](#)
- file [hash_fun.h](#)
- file [hash_map](#)
- file [hash_set](#)
- file [backward/hashtable.h](#)
- file [stringstream](#)

3.2 include/x86_64-unknown-linux-gnu/bits/ Directory Reference

Directory dependency graph for include/x86_64-unknown-linux-gnu/bits/:



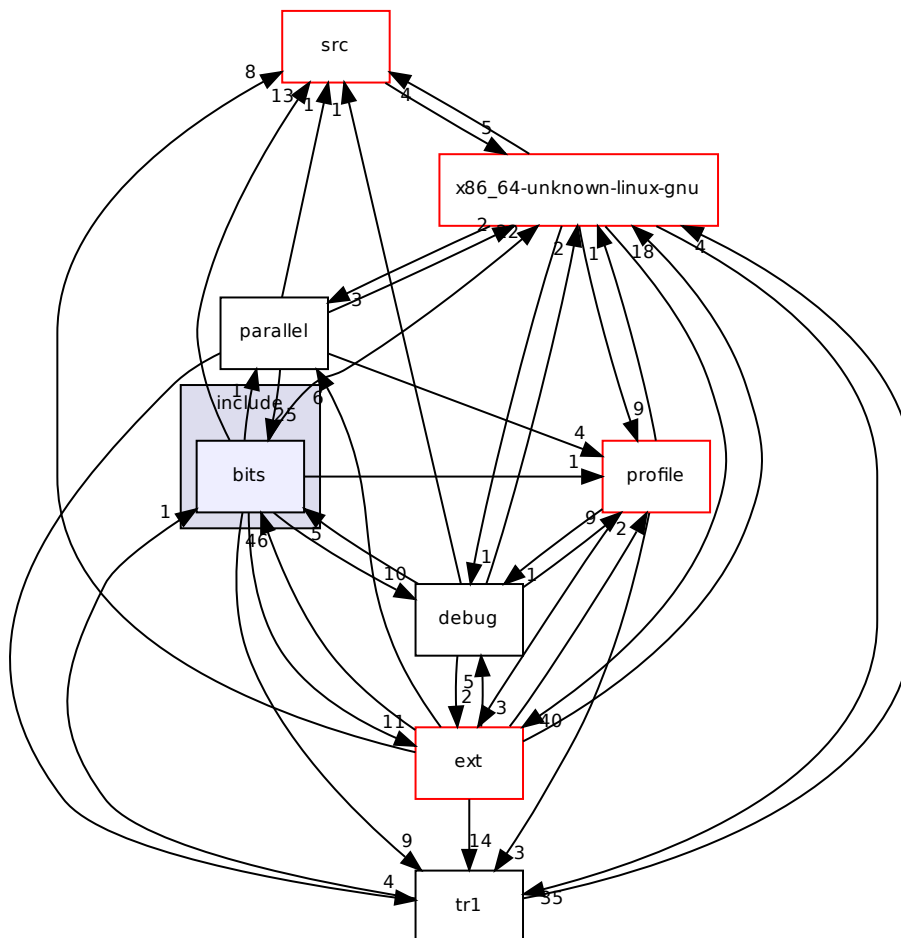
Files

- file [atomic_word.h](#)
- file [basic_file.h](#)
- file [c++allocator.h](#)

- file [c++config.h](#)
- file [c++io.h](#)
- file [c++locale.h](#)
- file [c++locale_internal.h](#)
- file [x86_64-unknown-linux-gnu/bits/compatibility.h](#)
- file [cpu_defines.h](#)
- file [ctype_base.h](#)
- file [ctype_inline.h](#)
- file [ctype_noninline.h](#)
- file [cxxabi_tweaks.h](#)
- file [error_constants.h](#)
- file [extc++.h](#)
- file **gthr-default.h**
- file **gthr-posix.h**
- file **gthr-single.h**
- file **gthr-tpf.h**
- file **gthr.h**
- file [messages_members.h](#)
- file [os_defines.h](#)
- file [stdc++.h](#)
- file [stdtr1c++.h](#)
- file [time_members.h](#)

3.3 include/bits/ Directory Reference

Directory dependency graph for include/bits/:



Files

- file [bits/algorithmfwd.h](#)
- file [allocator.h](#)
- file [atomic_0.h](#)

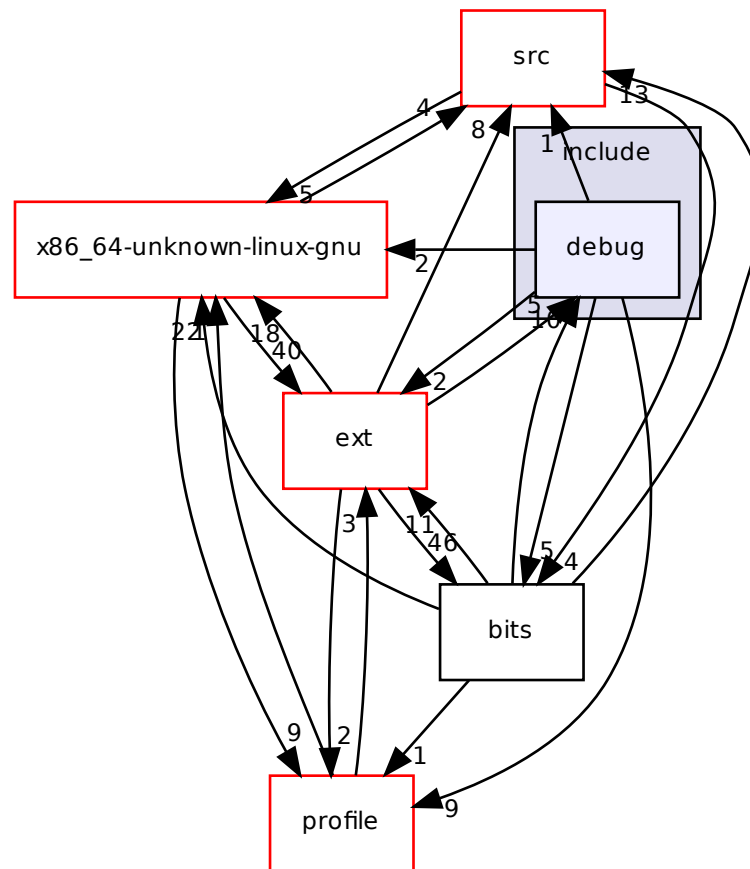
- file [atomic_2.h](#)
- file [atomic_base.h](#)
- file [basic_ios.h](#)
- file [basic_ios.tcc](#)
- file [basic_string.h](#)
- file [basic_string.tcc](#)
- file [boost_concept_check.h](#)
- file [c++0x_warning.h](#)
- file [char_traits.h](#)
- file [codecvt.h](#)
- file [concept_check.h](#)
- file [cpp_type_traits.h](#)
- file [cxxabi_forced.h](#)
- file [deque.tcc](#)
- file [exception_defines.h](#)
- file [exception_ptr.h](#)
- file [forward_list.h](#)
- file [forward_list.tcc](#)
- file [fstream.tcc](#)
- file [functexcept.h](#)
- file [functional_hash.h](#)
- file [gslice.h](#)
- file [gslice_array.h](#)
- file [hash_bytes.h](#)
- file [bits/hashtable.h](#)
- file [hashtable_policy.h](#)
- file [indirect_array.h](#)
- file [ios_base.h](#)
- file [istream.tcc](#)
- file [list.tcc](#)
- file [locale_classes.h](#)
- file [locale_classes.tcc](#)
- file [locale_facets.h](#)
- file [locale_facets.tcc](#)
- file [locale_facets_nonio.h](#)
- file [locale_facets_nonio.tcc](#)
- file [localefwd.h](#)
- file [mask_array.h](#)
- file [move.h](#)
- file [nested_exception.h](#)
- file [ostream.tcc](#)
- file [ostream_insert.h](#)
- file [postypes.h](#)

- file [random.h](#)
- file [random.tcc](#)
- file [range_access.h](#)
- file [regex.h](#)
- file [regex_compiler.h](#)
- file [regex_constants.h](#)
- file [regex_cursor.h](#)
- file [regex_error.h](#)
- file [regex_grep_matcher.h](#)
- file [regex_grep_matcher.tcc](#)
- file [regex_nfa.h](#)
- file [regex_nfa.tcc](#)
- file [shared_ptr.h](#)
- file [shared_ptr_base.h](#)
- file [slice_array.h](#)
- file [sstream.tcc](#)
- file [stl_algo.h](#)
- file [stl_algobase.h](#)
- file [stl_bvector.h](#)
- file [stl_construct.h](#)
- file [stl_deque.h](#)
- file [stl_function.h](#)
- file [stl_heap.h](#)
- file [stl_iterator.h](#)
- file [stl_iterator_base_funcs.h](#)
- file [stl_iterator_base_types.h](#)
- file [stl_list.h](#)
- file [stl_map.h](#)
- file [stl_multimap.h](#)
- file [stl_multiset.h](#)
- file [stl_numeric.h](#)
- file [stl_pair.h](#)
- file [stl_queue.h](#)
- file [stl_raw_storage_iter.h](#)
- file [stl_relops.h](#)
- file [stl_set.h](#)
- file [stl_stack.h](#)
- file [stl_tempbuf.h](#)
- file [stl_tree.h](#)
- file [stl_uninitialized.h](#)
- file [stl_vector.h](#)
- file [stream_iterator.h](#)
- file [streambuf.tcc](#)

- file [streambuf_iterator.h](#)
- file [stringfwd.h](#)
- file [unique_ptr.h](#)
- file [unordered_map.h](#)
- file [unordered_set.h](#)
- file [valarray_after.h](#)
- file [valarray_array.h](#)
- file [valarray_array.tcc](#)
- file [valarray_before.h](#)
- file [vector.tcc](#)

3.4 include/debug/ Directory Reference

Directory dependency graph for include/debug/:



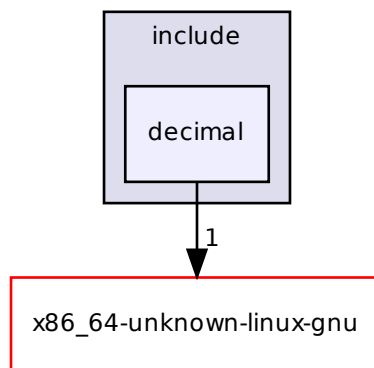
Files

- file [debug/bitset](#)
- file [debug.h](#)
- file [debug/deque](#)

- file [formatter.h](#)
- file [debug/forward_list](#)
- file [functions.h](#)
- file [debug/list](#)
- file [macros.h](#)
- file [debug/map](#)
- file [debug/map.h](#)
- file [debug/multimap.h](#)
- file [debug/multiset.h](#)
- file [safe_base.h](#)
- file [safe_iterator.h](#)
- file [safe_iterator.tcc](#)
- file [safe_sequence.h](#)
- file [safe_sequence.tcc](#)
- file [debug/set](#)
- file [debug/set.h](#)
- file [debug/string](#)
- file [debug/unordered_map](#)
- file [debug/unordered_set](#)
- file [debug/vector](#)

3.5 include/decimal/ Directory Reference

Directory dependency graph for include/decimal/:

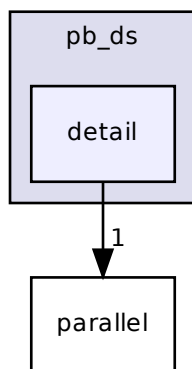


Files

- file [decimal](#)

3.6 include/ext/pb_ds/detail/ Directory Reference

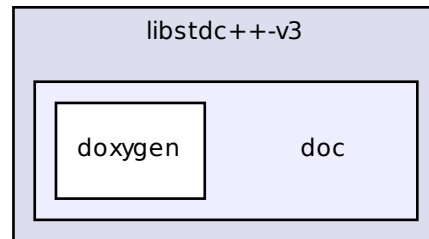
Directory dependency graph for include/ext/pb_ds/detail/:

**Files**

- file [basic_types.hpp](#)
- file [cond_dealtor.hpp](#)
- file [constructors_destructor_fn_imps.hpp](#)
- file [container_base_dispatch.hpp](#)
- file [debug_map_base.hpp](#)
- file [priority_queue_base_dispatch.hpp](#)
- file [standard_policies.hpp](#)
- file [tree_trace_base.hpp](#)
- file [type_utils.hpp](#)
- file [types_traits.hpp](#)

3.7 /mnt/share/src/gcc.git-trunk/libstdc++-v3/doc/ Directory Reference

Directory dependency graph for /mnt/share/src/gcc.git-trunk/libstdc++-v3/doc/:



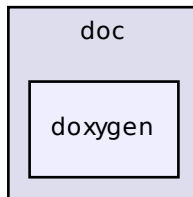
Directories

- directory [doxygen](#)

3.8 /mnt/share/src/gcc.git-trunk/libstdc++-v3/doc/doxygen/ Directory Reference 344

3.8 /mnt/share/src/gcc.git-trunk/libstdc++-v3/doc/doxygen/ Directory Reference

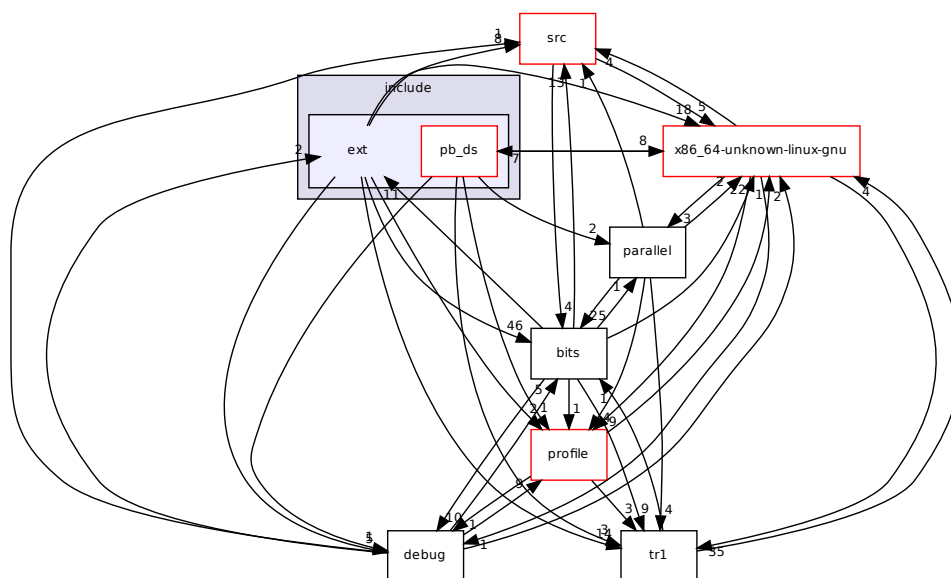
Directory dependency graph for /mnt/share/src/gcc.git-trunk/libstdc++-v3/doc/doxygen/:



Files

- file `doxygroups.cc`

Directory dependency graph for include/ext/:



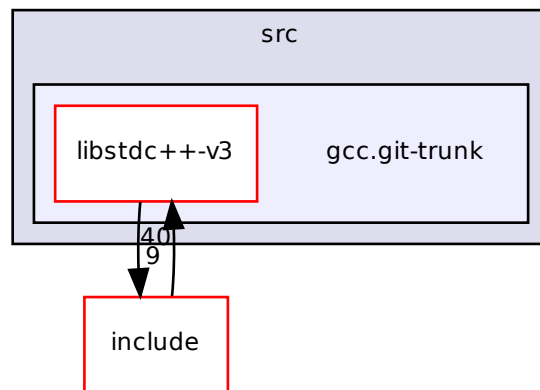
- directory `pb_ds`

- file [ext/algorithm](#)
- file [array_allocator.h](#)
- file [atomicity.h](#)
- file [bitmap_allocator.h](#)
- file [cast.h](#)
- file [codecvt_specializations.h](#)
- file [concurrency.h](#)
- file [debug_allocator.h](#)
- file [enc_filebuf.h](#)
- file [extptr_allocator.h](#)

- file [ext/functional](#)
- file [ext/iterator](#)
- file [malloc_allocator.h](#)
- file [ext/memory](#)
- file [mt_allocator.h](#)
- file [new_allocator.h](#)
- file [ext/numeric](#)
- file [numeric_traits.h](#)
- file [pod_char_traits.h](#)
- file [pointer.h](#)
- file [pool_allocator.h](#)
- file [rb_tree](#)
- file [rc_string_base.h](#)
- file [rope](#)
- file [ropeimpl.h](#)
- file [slist](#)
- file [sso_string_base.h](#)
- file [stdio_filebuf.h](#)
- file [stdio_sync_filebuf.h](#)
- file [string_conversions.h](#)
- file [throw_allocator.h](#)
- file [type_traits.h](#)
- file [typelist.h](#)
- file [vstring.h](#)
- file [vstring.tcc](#)
- file [vstring_fwd.h](#)
- file [vstring_util.h](#)

3.10 /mnt/share/src/gcc.git-trunk/ Directory Reference

Directory dependency graph for /mnt/share/src/gcc.git-trunk/:

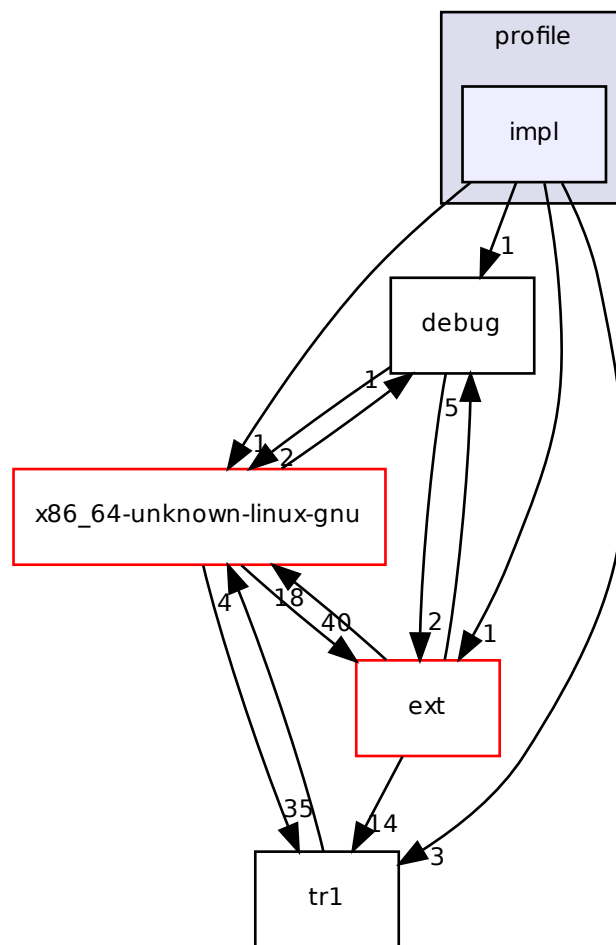


Directories

- directory [libstdc++-v3](#)

3.11 include/profile/impl/ Directory Reference

Directory dependency graph for include/profile/impl/:

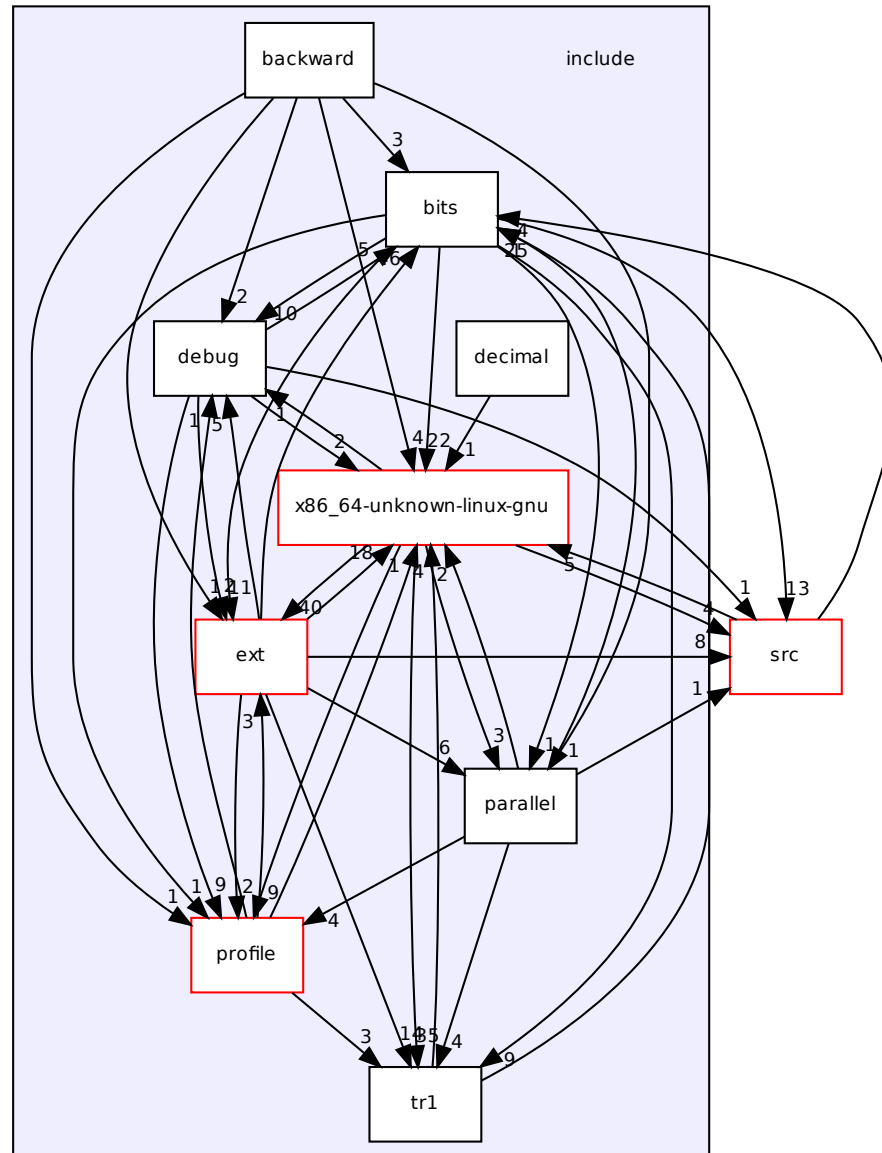


Files

- file [profiler.h](#)
- file [profiler_algos.h](#)
- file [profiler_container_size.h](#)
- file [profiler_hash_func.h](#)
- file [profiler_hashtable_size.h](#)
- file [profiler_list_to_slist.h](#)
- file [profiler_list_to_vector.h](#)
- file [profiler_map_to_unordered_map.h](#)
- file [profiler_node.h](#)
- file [profiler_state.h](#)
- file [profiler_trace.h](#)
- file [profiler_vector_size.h](#)
- file [profiler_vector_to_list.h](#)

3.12 include/ Directory Reference

Directory dependency graph for include/:



Directories

- directory [backward](#)
- directory [bits](#)
- directory [debug](#)
- directory [decimal](#)
- directory [ext](#)
- directory [parallel](#)
- directory [profile](#)
- directory [tr1](#)
- directory [x86_64-unknown-linux-gnu](#)

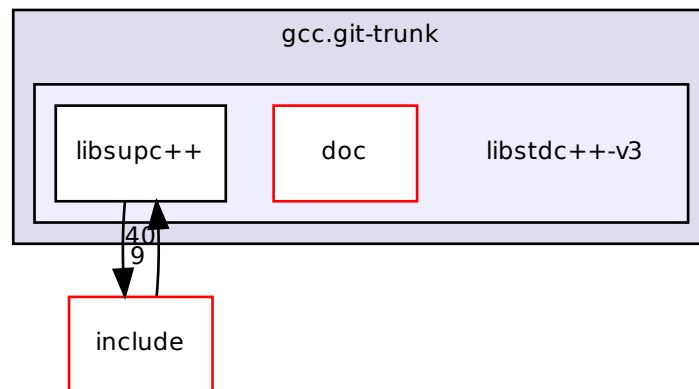
Files

- file [algorithm](#)
- file [array](#)
- file [atomic](#)
- file [bitset](#)
- file [cassert](#)
- file [ccomplex](#)
- file [cctype](#)
- file [cerrno](#)
- file [cfenv](#)
- file [cfloat](#)
- file [chrono](#)
- file [cinttypes](#)
- file [ciso646](#)
- file [climits](#)
- file [clocale](#)
- file [cmath](#)
- file [complex](#)
- file [complex.h](#)
- file [condition_variable](#)
- file [csetjmp](#)
- file [csignal](#)
- file [cstdarg](#)
- file [cstdbool](#)
- file [cstddef](#)
- file [cstdint](#)
- file [cstdio](#)
- file [cstdlib](#)
- file [cstring](#)
- file [ctgmath](#)

- file [ctime](#)
- file [cwchar](#)
- file [cwctype](#)
- file [deque](#)
- file [fenv.h](#)
- file [forward_list](#)
- file [fstream](#)
- file [functional](#)
- file [future](#)
- file **[gstdint.h](#)**
- file [iomanip](#)
- file [ios](#)
- file [iosfwd](#)
- file [iostream](#)
- file [istream](#)
- file [iterator](#)
- file [limits](#)
- file [list](#)
- file [locale](#)
- file [map](#)
- file [memory](#)
- file [mutex](#)
- file [numeric](#)
- file [ostream](#)
- file [queue](#)
- file [random](#)
- file [ratio](#)
- file [regex](#)
- file [set](#)
- file [sstream](#)
- file [stack](#)
- file [stdexcept](#)
- file [streambuf](#)
- file [string](#)
- file [system_error](#)
- file [tgmath.h](#)
- file [thread](#)
- file [tuple](#)
- file [type_traits](#)
- file [typeindex](#)
- file [unordered_map](#)
- file [unordered_set](#)
- file [utility](#)
- file [valarray](#)
- file [vector](#)

3.13 /mnt/share/src/gcc.git-trunk/libstdc++-v3/ Directory Reference

Directory dependency graph for /mnt/share/src/gcc.git-trunk/libstdc++-v3/:

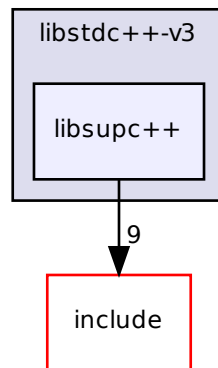


Directories

- directory [doc](#)
- directory [libsupc++](#)

3.14 /mnt/share/src/gcc.git-trunk/libstdc++-v3/libsupc++/ Directory Reference

Directory dependency graph for /mnt/share/src/gcc.git-trunk/libstdc++-v3/libsupc++/:

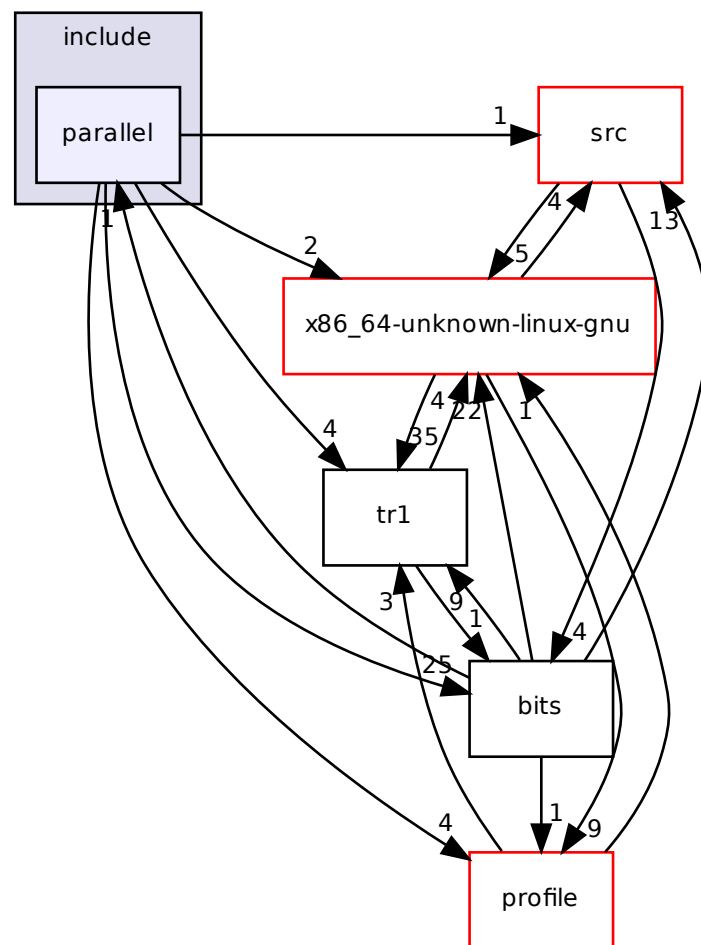


Files

- file [cxxabi.h](#)
- file [exception](#)
- file [initializer_list](#)
- file [new](#)
- file [typeinfo](#)

3.15 include/parallel/ Directory Reference

Directory dependency graph for include/parallel/:

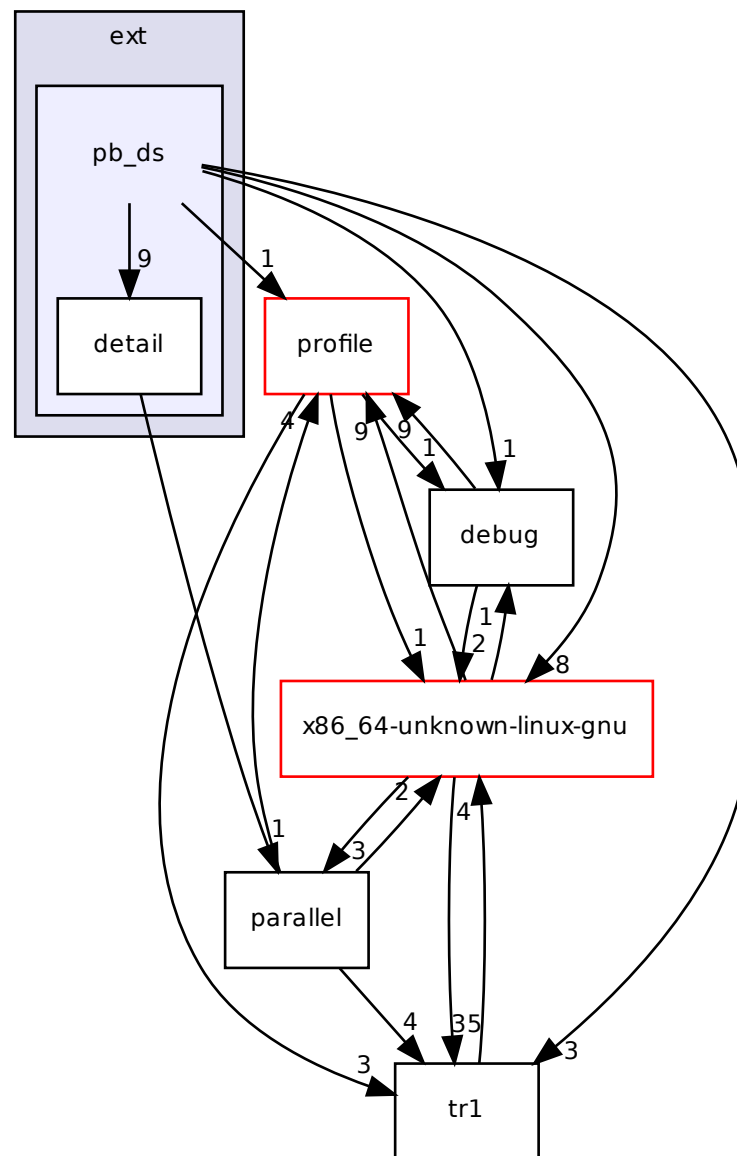


Files

- file [algo.h](#)
- file [algbase.h](#)
- file [parallel/algorithm](#)
- file [parallel/algorithmfwd.h](#)
- file [balanced_quicksort.h](#)
- file [parallel/base.h](#)
- file [basic_iterator.h](#)
- file [checkers.h](#)
- file [parallel/compatibility.h](#)
- file [compiletime_settings.h](#)
- file [equally_split.h](#)
- file [features.h](#)
- file [find.h](#)
- file [find_selectors.h](#)
- file [for_each.h](#)
- file [for_each_selectors.h](#)
- file [iterator.h](#)
- file [list_partition.h](#)
- file [losertree.h](#)
- file [merge.h](#)
- file [multiseq_selection.h](#)
- file [multiway_merge.h](#)
- file [multiway_mergesort.h](#)
- file [parallel/numeric](#)
- file [numericfwd.h](#)
- file [omp_loop.h](#)
- file [omp_loop_static.h](#)
- file [par_loop.h](#)
- file [parallel.h](#)
- file [partial_sum.h](#)
- file [partition.h](#)
- file [queue.h](#)
- file [quicksort.h](#)
- file [random_number.h](#)
- file [random_shuffle.h](#)
- file [search.h](#)
- file [set_operations.h](#)
- file [settings.h](#)
- file [sort.h](#)
- file [tags.h](#)
- file [types.h](#)
- file [unique_copy.h](#)
- file [workstealing.h](#)

3.16 include/ext/pb_ds/ Directory Reference

Directory dependency graph for include/ext/pb_ds/:



Directories

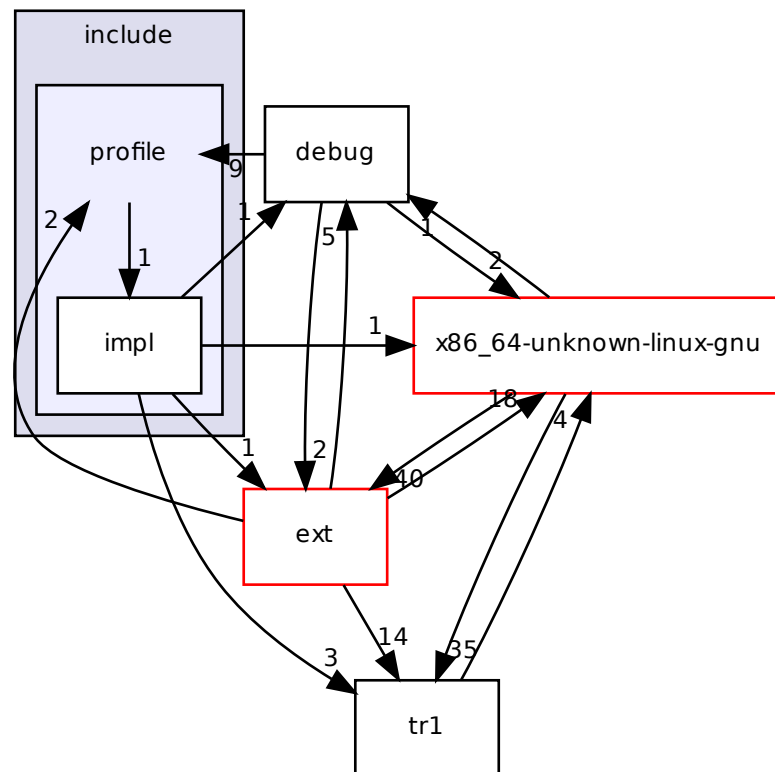
- directory [detail](#)

Files

- file [assoc_container.hpp](#)
- file [exception.hpp](#)
- file [hash_policy.hpp](#)
- file [list_update_policy.hpp](#)
- file [priority_queue.hpp](#)
- file [tag_and_trait.hpp](#)
- file [tree_policy.hpp](#)
- file [trie_policy.hpp](#)

3.17 include/profile/ Directory Reference

Directory dependency graph for include/profile/:



Directories

- directory [impl](#)

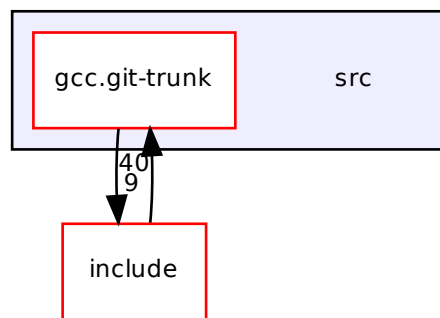
Files

- file [profile/base.h](#)

- file [profile/bitset](#)
- file [profile/deque](#)
- file [profile/forward_list](#)
- file [iterator_tracker.h](#)
- file [profile/list](#)
- file [profile/map](#)
- file [profile/map.h](#)
- file [profile/multimap.h](#)
- file [profile/multiset.h](#)
- file [profile/set](#)
- file [profile/set.h](#)
- file [profile/unordered_map](#)
- file [profile/unordered_set](#)
- file [profile/vector](#)

3.18 /mnt/share/src/ Directory Reference

Directory dependency graph for /mnt/share/src/:

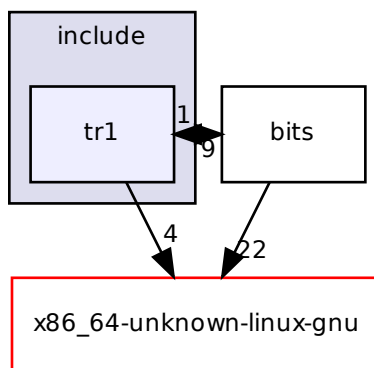


Directories

- directory [gcc.git-trunk](#)

3.19 include/tr1/ Directory Reference

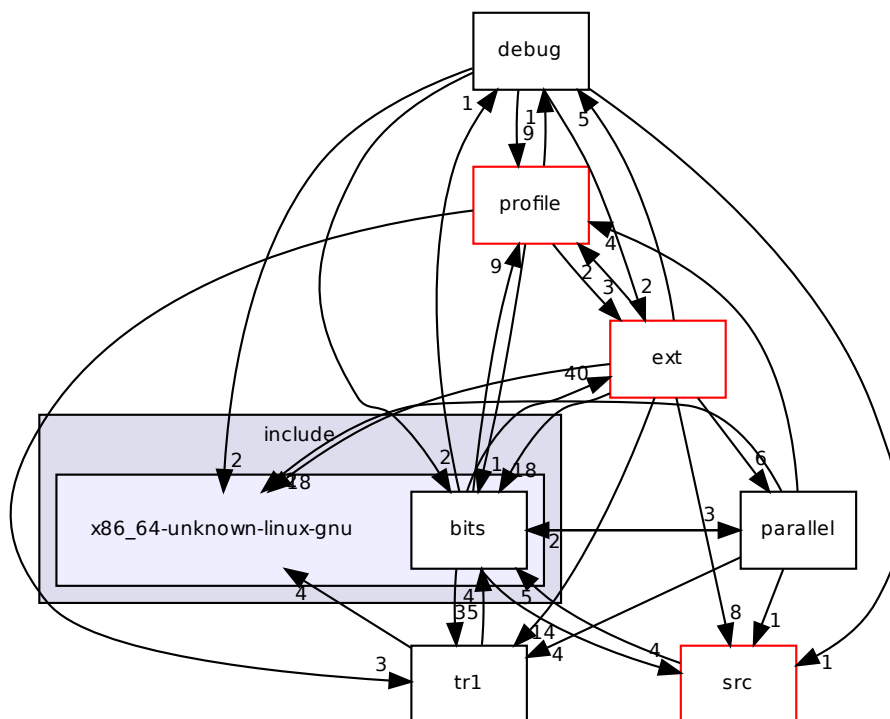
Directory dependency graph for include/tr1/:



Files

- file [tr1/complex](#)
- file [tr1/cctype](#)
- file [tr1/cfenv](#)
- file [tr1/cfloat](#)
- file [tr1/cinttypes](#)
- file [tr1/climits](#)
- file [tr1/cmath](#)
- file [tr1/complex](#)
- file [tr1/cstdarg](#)
- file [tr1/cstdbool](#)
- file [tr1/cstdint](#)
- file [tr1/cstdio](#)
- file [tr1/cstdlib](#)
- file [tr1/ctgmath](#)
- file [tr1/ctime](#)
- file [tr1/cwchar](#)
- file [tr1/cwctype](#)

3.20 include/x86_64-unknown-linux-gnu/ Directory Reference



Directories

4 Namespace Documentation

GNU extensions for public use.

Namespaces

- namespace [__detail](#)
- namespace [typelist](#)

Classes

- struct [__common_pool_policy](#)
Policy for shared __pool objects.
- class [__mt_alloc](#)
This is a fixed size (power of 2) allocator which - when compiled with thread support - will maintain one freelist per size per thread plus a global one. Steps are taken to limit the per thread freelist sizes (by returning excess back to the global list). Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt12ch32.html>.
- class [__mt_alloc_base](#)
Base class for _Tp dependent member functions.
- struct [__per_type_pool_policy](#)
Policy for individual __pool objects.
- class [__pool< false >](#)
Specialization for single thread.
- class [__pool< true >](#)
Specialization for thread enabled, via gthreads.h.
- class [__pool_alloc](#)
Allocator using a memory pool with a single lock.
- class [__pool_alloc_base](#)
Base class for __pool_alloc.
- struct [__pool_base](#)
Base class for pool object.
- class [__rc_string_base](#)
- class [__scoped_lock](#)
Scoped lock idiom.
- class [__versa_string](#)

Template class `__versa_string`.

Data structure managing sequences of characters and character-like objects.

- struct `_Caster`
- struct `_Char_types`

Mapping from character type to associated types.

- class `_ExtPtr_allocator`

An example allocator which uses a non-standard pointer type.

This allocator specifies that containers use a 'relative pointer' as it's pointer type. (See [ext/pointer.h](#)) Memory allocation in this example is still performed using `std::allocator`.

- struct `_Invalid_type`
- class `_Pointer_adapter`
- class `_Relative_pointer_impl`

A storage policy for use with `_Pointer_adapter<>` which stores the pointer's address as an offset value which is relative to its own address.

- class `_Relative_pointer_impl< const_Tp >`
- class `_Std_pointer_impl`

A storage policy for use with `_Pointer_adapter<>` which yields a standard pointer.

- struct `_Unqualified_type`
- struct `annotate_base`

Base class for checking address and label information about allocations. Create a [std::map](#) between the allocated address (`void*`) and a datum for annotations, which are a pair of numbers corresponding to label and allocated size.

- class `array_allocator`

An allocator that uses previously allocated memory. This memory can be externally, globally, or otherwise allocated.

- class `array_allocator_base`

Base class.

- class `binary_compose`

An *SGI extension*.

- class `bitmap_allocator`

Bitmap Allocator, primary template.

- struct `char_traits`

Base class used to implement `std::char_traits`.

- struct `character`
A POD class that serves as a character abstraction class.
- struct `condition_base`
Base struct for condition policy.
- struct `constant_binary_fun`
An SGI extension .
- struct `constant_unary_fun`
An SGI extension .
- struct `constant_void_fun`
An SGI extension .
- class `debug_allocator`
*A meta-allocator with debugging bits, as per [20.4].
This is precisely the allocator defined in the C++ Standard.*
 - all allocation calls operator new
 - all deallocation calls operator delete.
- class `enc_filebuf`
class `enc_filebuf`.
- struct `encoding_char_traits`
`encoding_char_traits`
- class `encoding_state`
Extension to use iconv for dealing with character encodings.
- struct `forced_error`
Thrown by exception safety machinery.
- class `free_list`
The free list class for managing chunks of memory to be given to and returned by the `bitmap_allocator`.
- class `hash_map`
- class `hash_multimap`
- class `hash_multiset`
- class `hash_set`
- struct `limit_condition`

Base class for incremental control and throw.

- class `malloc_allocator`

An allocator that uses malloc.

This is precisely the allocator defined in the C++ Standard.

- all allocation calls `malloc`
- all deallocation calls `free`.

- class `new_allocator`

An allocator that uses global new, as per [20.4].

This is precisely the allocator defined in the C++ Standard.

- all allocation calls operator `new`
- all deallocation calls operator `delete`.

- struct `project1st`

An SGI extension .

- struct `project2nd`

An SGI extension .

- struct `random_condition`

Base class for random probability control and throw.

- struct `rb_tree`

- class `recursive_init_error`

Exception thrown by `__cxa_guard_acquire`.

6.7[smt.dcl]/4: If control re-enters the declaration (recursively) while the object is being initialized, the behavior is undefined.

- class `rope`

- struct `select1st`

An SGI extension .

- struct `select2nd`

An SGI extension .

- class `slist`

- class `stdio_filebuf`

Provides a layer of compatibility for C/POSIX.

This GNU extension provides extensions for working with standard C `FILE`'s and POSIX file descriptors. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.*

- class `stdio_sync_filebuf`
*Provides a layer of compatibility for C.
This GNU extension provides extensions for working with standard C FILE*'s. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.*
- class `subtractive_rng`
- struct `temporary_buffer`
- class `throw_allocator_base`
*Allocator class with logging and exception generation control. Intended to be used as an `allocator_type` in templated code.
Note: Deallocate not allowed to throw.*
- struct `throw_allocator_limit`
Allocator throwing via limit condition.
- struct `throw_allocator_random`
Allocator throwing via random condition.
- struct `throw_value_base`
Class with exception generation control. Intended to be used as a `value_type` in templated code.
- struct `throw_value_limit`
Type throwing via limit condition.
- struct `throw_value_random`
Type throwing via random condition.
- class `unary_compose`
An SGI extension .

Typedefs

- typedef void(* **__destroy_handler**)(void *)
- typedef `__versa_string`< char, `std::char_traits`< char >, `std::allocator`< char >, `__rc_string_base` > **__rc_string**
- typedef `__vstring` **__sso_string**
- typedef `__versa_string`< char16_t, `std::char_traits`< char16_t >, `std::allocator`< char16_t >, `__rc_string_base` > **__u16rc_string**
- typedef `__u16vstring` **__u16sso_string**
- typedef `__versa_string`< char16_t > **__u16vstring**

- typedef `__versa_string`< `char32_t`, `std::char_traits`< `char32_t`>, `std::allocator`< `char32_t`>, `__rc_string_base`> `__u32rc_string`
- typedef `__u32vstring` `__u32sso_string`
- typedef `__versa_string`< `char32_t`> `__u32vstring`
- typedef `__versa_string`< `char`> `__vstring`
- typedef `__versa_string`< `wchar_t`, `std::char_traits`< `wchar_t`>, `std::allocator`< `wchar_t`>, `__rc_string_base`> `__wrc_string`
- typedef `__wvstring` `__wsso_string`
- typedef `__versa_string`< `wchar_t`> `__wvstring`
- typedef `rope`< `char`> `crope`
- typedef `rope`< `wchar_t`> `wrope`

Enumerations

- enum { `_S_num_primes` }
- enum `_Lock_policy` { `_S_single`, `_S_mutex`, `_S_atomic` }

Functions

- static void `__atomic_add` (volatile `_Atomic_word` *`__mem`, int `__val`)
- static void `__atomic_add_single` (`_Atomic_word` *`__mem`, int `__val`)
- static `_Atomic_word` `__attribute__` ((`__unused__`)) `__exchange_and_add_dispatch`(`_Atomic_word` *`__mem`)
- template<class `_Tp`>
void `__aux_require_boolean_expr` (const `_Tp` &`__t`)
- template<typename `_ToType`, typename `_FromType`>
`_ToType` `__const_pointer_cast` (const `_FromType` &`__arg`)
- template<typename `_ToType`, typename `_FromType`>
`_ToType` `__const_pointer_cast` (`_FromType` *`__arg`)
- template<typename `_InputIterator`, typename `_Size`, typename `_OutputIterator`>
`pair`< `_InputIterator`, `_OutputIterator`> `__copy_n` (`_InputIterator` `__first`, `_Size` `__count`, `_OutputIterator` `__result`, `input_iterator_tag`)
- template<typename `_RAIterator`, typename `_Size`, typename `_OutputIterator`>
`pair`< `_RAIterator`, `_OutputIterator`> `__copy_n` (`_RAIterator` `__first`, `_Size` `__count`, `_OutputIterator` `__result`, `random_access_iterator_tag`)
- template<typename `_InputIterator`, typename `_Distance`>
void `__distance` (`_InputIterator` `__first`, `_InputIterator` `__last`, `_Distance` &`__n`, `std::input_iterator_tag`)
- template<typename `_RandomAccessIterator`, typename `_Distance`>
void `__distance` (`_RandomAccessIterator` `__first`, `_RandomAccessIterator` `__last`, `_Distance` &`__n`, `std::random_access_iterator_tag`)
- template<typename `_ToType`, typename `_FromType`>
`_ToType` `__dynamic_pointer_cast` (const `_FromType` &`__arg`)

- `template<typename _ToType, typename _FromType >`
`_ToType __dynamic_pointer_cast (_FromType * __arg)`
- `void __error_type_must_be_a_signed_integer_type ()`
- `void __error_type_must_be_an_integer_type ()`
- `void __error_type_must_be_an_unsigned_integer_type ()`
- `static _Atomic_word __exchange_and_add (volatile _Atomic_word * __mem,`
`int __val)`
- `static _Atomic_word __exchange_and_add_single (_Atomic_word * __mem,`
`int __val)`
- `else return __exchange_and_add_single (__mem, __val)`
- `template<class _Concept >`
`void __function_requires ()`
- `template<typename _Type >`
`bool __is_null_pointer (_Type * __ptr)`
- `template<typename _Type >`
`bool __is_null_pointer (_Type)`
- `int __lexicographical_compare_3way (const unsigned char * __first1, const un-`
`signed char * __last1, const unsigned char * __first2, const unsigned char * __-`
`last2)`
- `int __lexicographical_compare_3way (const char * __first1, const char * __-`
`last1, const char * __first2, const char * __last2)`
- `template<typename _InputIterator1, typename _InputIterator2 >`
`int __lexicographical_compare_3way (_InputIterator1 __first1, _InputIterator1`
`__last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _Tp >`
`const _Tp & __median (const _Tp & __a, const _Tp & __b, const _Tp & __c)`
- `template<typename _Tp, typename _Compare >`
`const _Tp & __median (const _Tp & __a, const _Tp & __b, const _Tp & __c,`
`_Compare __comp)`
- `crope::reference __mutable_reference_at (crope & __c, size_t __i)`
- `template<typename _Tp, typename _Integer, typename _MonoidOperation >`
`_Tp __power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`
- `template<typename _Tp, typename _Integer >`
`_Tp __power (_Tp __x, _Integer __n)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _-`
`RandomNumberGenerator, typename _Distance >`
`_RandomAccessIterator __random_sample (_InputIterator __first, _-`
`InputIterator __last, _RandomAccessIterator __out, _RandomNumberGenerator`
`& __rand, const _Distance __n)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Distance >`
`_RandomAccessIterator __random_sample (_InputIterator __first, _-`
`InputIterator __last, _RandomAccessIterator __out, const _Distance __n)`
- `template<typename _ToType, typename _FromType >`
`_ToType __reinterpret_pointer_cast (const _FromType & __arg)`

- `template<typename _ToType, typename _FromType >`
`_ToType __reinterpret_pointer_cast (_FromType * __arg)`
- `_Slist_node_base * __slist_make_link (_Slist_node_base * __prev_node, _Slist_node_base * __new_node)`
- `_Slist_node_base * __slist_previous (_Slist_node_base * __head, const _Slist_node_base * __node)`
- `const _Slist_node_base * __slist_previous (const _Slist_node_base * __head, const _Slist_node_base * __node)`
- `_Slist_node_base * __slist_reverse (_Slist_node_base * __node)`
- `size_t __slist_size (_Slist_node_base * __node)`
- `void __slist_splice_after (_Slist_node_base * __pos, _Slist_node_base * __before_first, _Slist_node_base * __before_last)`
- `void __slist_splice_after (_Slist_node_base * __pos, _Slist_node_base * __head)`
- `template<typename _ToType, typename _FromType >`
`_ToType __static_pointer_cast (const _FromType & __arg)`
- `template<typename _ToType, typename _FromType >`
`_ToType __static_pointer_cast (_FromType * __arg)`
- `size_t __stl_hash_string (const char * __s)`
- `unsigned long __stl_next_prime (unsigned long __n)`
- `template<typename _TRet, typename _Ret = _TRet, typename _CharT, typename... _Base>`
`_Ret __stoa (_TRet (* __convf) (const _CharT *, _CharT **, _Base...), const char * __name, const _CharT * __str, std::size_t * __idx, _Base... __base)`
- `void __throw_concurrency_lock_error ()`
- `void __throw_concurrency_unlock_error ()`
- `void __throw_forced_error ()`
- `template<typename _String, typename _CharT = typename _String::value_type>`
`_String __to_xstring (int (* __convf) (_CharT *, std::size_t, const _CharT *, __builtin_va_list), std::size_t __n, const _CharT * __fmt,...)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter >`
`pair< _InputIter, _ForwardIter > __uninitialized_copy_n (_InputIter __first, _Size __count, _ForwardIter __result, std::input_iterator_tag)`
- `template<typename _RandomAccessIter, typename _Size, typename _ForwardIter >`
`pair< _RandomAccessIter, _ForwardIter > __uninitialized_copy_n (_RandomAccessIter __first, _Size __count, _ForwardIter __result, std::random_access_iterator_tag)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter >`
`pair< _InputIter, _ForwardIter > __uninitialized_copy_n (_InputIter __first, _Size __count, _ForwardIter __result)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Allocator >`
`pair< _InputIter, _ForwardIter > __uninitialized_copy_n_a (_InputIter __first, _Size __count, _ForwardIter __result, _Allocator __alloc)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Tp >`
`pair< _InputIter, _ForwardIter > __uninitialized_copy_n_a (_InputIter __first, _Size __count, _ForwardIter __result, std::allocator< _Tp >)`

- `void __verbose_terminate_handler ()`
- `size_t __Bit_scan_forward (size_t __num)`
- `template<typename _ForwardIterator, typename _Allocator >`
`void __Destroy_const (_ForwardIterator __first, _ForwardIterator __last, _-`
`Allocator __alloc)`
- `template<typename _ForwardIterator, typename _Tp >`
`void __Destroy_const (_ForwardIterator __first, _ForwardIterator __last, alloca-`
`tor< _Tp >)`
- `template<class _CharT, class _Traits >`
`void __Rope_fill (basic_ostream< _CharT, _Traits > &__o, size_t __n)`
- `template<class _CharT >`
`bool __Rope_is_simple (_CharT *)`
- `bool __Rope_is_simple (char *)`
- `bool __Rope_is_simple (wchar_t *)`
- `template<class _Rope_iterator >`
`void __Rope_rotate (_Rope_iterator __first, _Rope_iterator __middle, _Rope_-`
`iterator __last)`
- `template<class _CharT >`
`void __S_cond_store_eos (_CharT &)`
- `void __S_cond_store_eos (char &__c)`
- `void __S_cond_store_eos (wchar_t &__c)`
- `template<class _CharT >`
`_CharT __S_eos (_CharT *)`
- `bool __S_is_basic_char_type (wchar_t *)`
- `template<class _CharT >`
`bool __S_is_basic_char_type (_CharT *)`
- `bool __S_is_basic_char_type (char *)`
- `bool __S_is_one_byte_char_type (char *)`
- `template<class _CharT >`
`bool __S_is_one_byte_char_type (_CharT *)`
- `template<class _Operation1, class _Operation2 >`
`unary_compose< _Operation1, _Operation2 > compose1 (const _Operation1`
`&__fn1, const _Operation2 &__fn2)`
- `template<class _Operation1, class _Operation2, class _Operation3 >`
`binary_compose< _Operation1, _Operation2, _Operation3 > compose2 (const`
`_Operation1 &__fn1, const _Operation2 &__fn2, const _Operation3 &__fn3)`
- `template<class _Result >`
`constant_void_fun< _Result > constant0 (const _Result &__val)`
- `template<class _Result >`
`constant_unary_fun< _Result, _Result > constant1 (const _Result &__val)`
- `template<class _Result >`
`constant_binary_fun< _Result, _Result, _Result > constant2 (const _Result &_-`
`__val)`

- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`
`pair< _InputIterator, _OutputIterator > copy_n (_InputIterator __first, _Size _`
`_count, _OutputIterator __result)`
- `template<typename _InputIterator, typename _Tp, typename _Size >`
`void count (_InputIterator __first, _InputIterator __last, const _Tp &__value,`
`_Size &__n)`
- `template<typename _InputIterator, typename _Predicate, typename _Size >`
`void count_if (_InputIterator __first, _InputIterator __last, _Predicate __pred,`
`_Size &__n)`
- `template<typename _InputIterator, typename _Distance >`
`void distance (_InputIterator __first, _InputIterator __last, _Distance &__n)`
- `template<class _Tp >`
`_Tp identity_element (std::plus< _Tp >)`
- `template<class _Tp >`
`_Tp identity_element (std::multiplies< _Tp >)`
- `static _Atomic_word int __val if (__pthread_active_p()) return __exchange_-`
`and_add(__mem`
- `template<typename _ForwardIter, typename _Tp >`
`void iota (_ForwardIter __first, _ForwardIter __last, _Tp __value)`
- `template<typename _RandomAccessIterator >`
`bool is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _StrictWeakOrdering >`
`bool is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _`
`StrictWeakOrdering __comp)`
- `template<typename _ForwardIterator, typename _StrictWeakOrdering >`
`bool is_sorted (_ForwardIterator __first, _ForwardIterator __last, _`
`StrictWeakOrdering __comp)`
- `template<typename _ForwardIterator >`
`bool is_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _InputIterator1, typename _InputIterator2 >`
`int lexicographical_compare_3way (_InputIterator1 __first1, _InputIterator1 __`
`last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<class _Ret, class _Tp, class _Arg >`
`mem_fun1_t< _Ret, _Tp, _Arg > mem_fun1 (_Ret(_Tp::*__f)(_Arg))`
- `template<class _Ret, class _Tp, class _Arg >`
`mem_fun1_ref_t< _Ret, _Tp, _Arg > mem_fun1_ref (_Ret(_Tp::*__f)(_Arg))`
- `template<class _Value, class _HashFcn, class _EqualKey, class _Alloc >`
`bool operator!= (const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &_`
`_hs1, const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &_hs2)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`bool operator!= (const __normal_iterator< _IteratorL, _Container > &__lhs,`
`const __normal_iterator< _IteratorR, _Container > &__rhs)`

- `template<typename _Iterator, typename _Container >`
`bool operator!= (const __normal_iterator< _Iterator, _Container > &__lhs,`
`const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _Tp, typename _Array >`
`bool operator!= (const array_allocator< _Tp, _Array > &, const array_-`
`allocator< _Tp, _Array > &)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator!= (const bitmap_allocator< _Tp1 > &, const bitmap_allocator<`
`_Tp2 > &) throw ()`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,`
`typename > class _Base>`
`bool operator!= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__`
`lhs, const _CharT * __rhs)`
- `template<typename _Tp >`
`bool operator!= (const malloc_allocator< _Tp > &, const malloc_allocator<`
`_Tp > &)`
- `template<class _CharT, class _Alloc >`
`bool operator!= (const _Rope_const_iterator< _CharT, _Alloc > &__x, const`
`_Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<typename _Tp, typename _Poolp >`
`bool operator!= (const __mt_alloc< _Tp, _Poolp > &, const __mt_alloc< _Tp,`
`_Poolp > &)`
- `template<typename _Tp >`
`bool operator!= (const new_allocator< _Tp > &, const new_allocator< _Tp >`
`&)`
- `template<class _CharT, class _Alloc >`
`bool operator!= (const _Rope_char_ptr_proxy< _CharT, _Alloc > &__x, const`
`_Rope_char_ptr_proxy< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool operator!= (const _Rope_iterator< _CharT, _Alloc > &__x, const _`
`Rope_iterator< _CharT, _Alloc > &__y)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator!= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator!= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator!= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_-`
`adapter< _Tp2 > &__rhs)`
- `template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc >`
`bool operator!= (const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc >`
`&__hm1, const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm2)`
- `template<typename _Tp >`
`bool operator!= (const _Pointer_adapter< _Tp > &__lhs, int __rhs)`

- `template<class _CharT, class _Alloc >`
`bool operator!= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<typename _Tp >`
`bool operator!= (int __lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp >`
`bool operator!= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`
`bool operator!= (const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<typename _Tp >`
`bool operator!= (const __pool_alloc< _Tp > &, const __pool_alloc< _Tp > &)`
- `template<typename _Tp, typename _Cond >`
`bool operator!= (const throw_allocator_base< _Tp, _Cond > &, const throw_allocator_base< _Tp, _Cond > &)`
- `template<class _Tp, class _Alloc >`
`bool operator!= (const slist< _Tp, _Alloc > &__SL1, const slist< _Tp, _Alloc > &__SL2)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >`
`bool operator!= (const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator!= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator!= (const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All >`
`bool operator!= (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht1, const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht2)`
- `template<typename _Cond >`
`throw_value_base< _Cond > operator* (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `template<class _CharT, class _Alloc >`
`_Rope_const_iterator< _CharT, _Alloc > operator+ (const _Rope_const_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `template<typename _Iterator, typename _Container >`
`__normal_iterator< _Iterator, _Container > operator+ (typename __normal_iterator< _Iterator, _Container >::difference_type __n, const __normal_iterator< _Iterator, _Container > &__i)`

- `template<class _CharT, class _Alloc >`
`_Rope_const_iterator< _CharT, _Alloc > operator+ (ptrdiff_t __n, const _Rope_const_iterator< _CharT, _Alloc > &__x)`
- `template<class _CharT, class _Alloc >`
`_Rope_iterator< _CharT, _Alloc > operator+ (const _Rope_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >`
`_Rope_iterator< _CharT, _Alloc > operator+ (ptrdiff_t __n, const _Rope_iterator< _CharT, _Alloc > &__x)`
- `template<class _CharT, class _Alloc >`
`rope< _CharT, _Alloc > operator+ (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc >`
`rope< _CharT, _Alloc > operator+ (const rope< _CharT, _Alloc > &__left, const _CharT *__right)`
- `template<class _CharT, class _Alloc >`
`rope< _CharT, _Alloc > operator+ (const rope< _CharT, _Alloc > &__left, _CharT __right)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (__versa_string< _CharT, _Traits, _Alloc, _Base > &&__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const _CharT *__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)`
- `template<typename _Cond >`
`throw_value_base< _Cond > operator+ (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`

`__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `_Base` > **operator+** (`_CharT` `__lhs`,
const `__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `_Base` > &`__rhs`)

- `template<typename _CharT, typename _Traits, typename _Alloc, template<typename, typename, typename> class _Base>`
`__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `_Base` > **operator+** (const `__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `_Base` > &`__lhs`, const `_CharT` *`__rhs`)
- `template<typename _CharT, typename _Traits, typename _Alloc, template<typename, typename, typename> class _Base>`
`__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `_Base` > **operator+** (const `__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `_Base` > &`__lhs`, `_CharT` `__rhs`)
- `template<typename _CharT, typename _Traits, typename _Alloc, template<typename, typename, typename> class _Base>`
`__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `_Base` > **operator+** (`__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `_Base` > &&`__lhs`, const `__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `_Base` > &`__rhs`)
- `template<typename _CharT, typename _Traits, typename _Alloc, template<typename, typename, typename> class _Base>`
`__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `_Base` > **operator+** (`__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `_Base` > &&`__lhs`, `__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `_Base` > &&`__rhs`)
- `template<typename _CharT, typename _Traits, typename _Alloc, template<typename, typename, typename> class _Base>`
`__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `_Base` > **operator+** (`_CharT` `__lhs`, `__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `_Base` > &&`__rhs`)
- `template<typename _CharT, typename _Traits, typename _Alloc, template<typename, typename, typename> class _Base>`
`__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `_Base` > **operator+** (`__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `_Base` > &&`__lhs`, `_CharT` `__rhs`)
- `template<class _CharT, class _Alloc>`
`rope`< `_CharT`, `_Alloc` > & **operator+=** (`rope`< `_CharT`, `_Alloc` > &`__left`, const `rope`< `_CharT`, `_Alloc` > &`__right`)
- `template<class _CharT, class _Alloc>`
`rope`< `_CharT`, `_Alloc` > & **operator+=** (`rope`< `_CharT`, `_Alloc` > &`__left`, const `_CharT` *`__right`)
- `template<class _CharT, class _Alloc>`
`rope`< `_CharT`, `_Alloc` > & **operator+=** (`rope`< `_CharT`, `_Alloc` > &`__left`, `_CharT` `__right`)
- `template<class _CharT, class _Alloc>`
`_Rope_const_iterator`< `_CharT`, `_Alloc` > **operator-** (const `_Rope_const_iterator`< `_CharT`, `_Alloc` > &`__x`, `ptrdiff_t` `__n`)
- `template<typename _IteratorL, typename _IteratorR, typename _Container>`
auto **operator-** (const `__normal_iterator`< `_IteratorL`, `_Container` > &`__lhs`, const `__normal_iterator`< `_IteratorR`, `_Container` > &`__rhs`)-> `decltype`(`__lhs`.`base`()-`__rhs`.`base`())

- `template<typename _Iterator, typename _Container >`
`__normal_iterator< _Iterator, _Container >::difference_type` **operator-** (const `__normal_iterator< _Iterator, _Container >` &__lhs, const `__normal_iterator< _Iterator, _Container >` &__rhs)
- `template<class _CharT, class _Alloc >`
`ptrdiff_t` **operator-** (const `_Rope_const_iterator< _CharT, _Alloc >` &__x, const `_Rope_const_iterator< _CharT, _Alloc >` &__y)
- `template<class _CharT, class _Alloc >`
`_Rope_iterator< _CharT, _Alloc >` **operator-** (const `_Rope_iterator< _CharT, _Alloc >` &__x, `ptrdiff_t` __n)
- `template<class _CharT, class _Alloc >`
`ptrdiff_t` **operator-** (const `_Rope_iterator< _CharT, _Alloc >` &__x, const `_Rope_iterator< _CharT, _Alloc >` &__y)
- `template<typename _Cond >`
`throw_value_base< _Cond >` **operator-** (const `throw_value_base< _Cond >` &__a, const `throw_value_base< _Cond >` &__b)
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`bool` **operator<** (const `__normal_iterator< _IteratorL, _Container >` &__lhs, const `__normal_iterator< _IteratorR, _Container >` &__rhs)
- `template<typename _Iterator, typename _Container >`
`bool` **operator<** (const `__normal_iterator< _Iterator, _Container >` &__lhs, const `__normal_iterator< _Iterator, _Container >` &__rhs)
- `template<class _CharT, class _Alloc >`
`bool` **operator<** (const `_Rope_const_iterator< _CharT, _Alloc >` &__x, const `_Rope_const_iterator< _CharT, _Alloc >` &__y)
- `template<class _CharT, class _Alloc >`
`bool` **operator<** (const `_Rope_iterator< _CharT, _Alloc >` &__x, const `_Rope_iterator< _CharT, _Alloc >` &__y)
- `template<typename V, typename I, typename S >`
`bool` **operator<** (const `character< V, I, S >` &lhs, const `character< V, I, S >` &rhs)
- `template<typename _Tp1, typename _Tp2 >`
`bool` **operator<** (const `_Pointer_adapter< _Tp1 >` &__lhs, `_Tp2` __rhs)
- `template<typename _Tp1, typename _Tp2 >`
`bool` **operator<** (`_Tp1` __lhs, const `_Pointer_adapter< _Tp2 >` &__rhs)
- `template<typename _Tp1, typename _Tp2 >`
`bool` **operator<** (const `_Pointer_adapter< _Tp1 >` &__lhs, const `_Pointer_adapter< _Tp2 >` &__rhs)
- `template<class _CharT, class _Alloc >`
`bool` **operator<** (const `rope< _CharT, _Alloc >` &__left, const `rope< _CharT, _Alloc >` &__right)
- `template<typename _CharT, typename _Traits, typename _Alloc, template<typename, typename, typename> class _Base>`
`bool` **operator<** (const `__versa_string< _CharT, _Traits, _Alloc, _Base >` &__lhs, const `_CharT *`__rhs)

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator< (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _Cond >`
`bool operator< (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `template<class _Tp, class _Alloc >`
`bool operator< (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator< (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _StoreT >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const _Pointer_adapter< _StoreT > &__p)`
- `template<class _CharT, class _Traits, class _Alloc >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__o, const rope< _CharT, _Alloc > &__r)`
- `std::ostream & operator<< (std::ostream &os, const annotate_base &__b)`
- `template<class _Tp, class _Alloc >`
`bool operator<= (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`bool operator<= (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)`
- `template<typename _Iterator, typename _Container >`
`bool operator<= (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator<= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator<= (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator<= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<class _CharT, class _Alloc >`
`bool operator<= (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`

- `template<class _CharT, class _Alloc >`
`bool operator<= (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator<= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator<= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<class _CharT, class _Alloc >`
`bool operator<= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<typename _Tp >`
`bool operator<= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator<= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`bool operator== (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)`
- `template<typename _Iterator, typename _Container >`
`bool operator== (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >`
`bool operator== (const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<class _CharT, class _Alloc >`
`bool operator== (const _Rope_char_ptr_proxy< _CharT, _Alloc > &__x, const _Rope_char_ptr_proxy< _CharT, _Alloc > &__y)`
- `template<class _Tp, class _Alloc >`
`bool operator== (const slist< _Tp, _Alloc > &__SL1, const slist< _Tp, _Alloc > &__SL2)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator== (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<class _Value, class _HashFn, class _EqualKey, class _Alloc >`
`bool operator== (const hash_set< _Value, _HashFn, _EqualKey, _Alloc > &__hs1, const hash_set< _Value, _HashFn, _EqualKey, _Alloc > &__hs2)`
- `template<typename _Tp, typename _Array >`
`bool operator== (const array_allocator< _Tp, _Array > &, const array_allocator< _Tp, _Array > &)`

- `template<typename _Tp >`
`bool operator== (const __pool_alloc< _Tp > &, const __pool_alloc< _Tp >`
`&)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator== (const bitmap_allocator< _Tp1 > &, const bitmap_allocator<`
`_Tp2 > &) throw ()`
- `template<class _CharT, class _Alloc >`
`bool operator== (const _Rope_iterator< _CharT, _Alloc > &__x, const _-`
`Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc >`
`bool operator== (const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc >`
`&__hm1, const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm2)`
- `template<typename _Tp, typename _Poolp >`
`bool operator== (const __mt_alloc< _Tp, _Poolp > &, const __mt_alloc< _-`
`Tp, _Poolp > &)`
- `template<typename _Tp >`
`bool operator== (const _Pointer_adapter< _Tp > &__lhs, int __rhs)`
- `template<typename V, typename I, typename S >`
`bool operator== (const character< V, I, S > &lhs, const character< V, I, S >`
`&rhs)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`
`bool operator== (const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc >`
`&__hs1, const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator== (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp, typename _Cond >`
`bool operator== (const throw_allocator_base< _Tp, _Cond > &, const throw_-`
`allocator_base< _Tp, _Cond > &)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,`
`typename > class _Base>`
`bool operator== (const __versa_string< _CharT, _Traits, _Alloc, _Base > &_-`
`__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _Tp >`
`bool operator== (int __lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<class _CharT, class _Alloc >`
`bool operator== (const rope< _CharT, _Alloc > &__left, const rope< _CharT,`
`_Alloc > &__right)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator== (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,`
`typename > class _Base>`
`bool operator== (const _CharT *__lhs, const __versa_string< _CharT, _Traits,`
`_Alloc, _Base > &__rhs)`

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator== (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`
- `template<typename _Cond >`
`bool operator== (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `template<typename _Tp >`
`bool operator== (const new_allocator< _Tp > &, const new_allocator< _Tp > &)`
- `template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All >`
`bool operator== (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht1, const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht2)`
- `template<class _CharT, class _Alloc >`
`bool operator== (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<typename _Tp >`
`bool operator== (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _CharT, template< typename, typename, typename > class _Base>`
`__enable_if< std::__is_char< _CharT >::__value, bool >::__type operator== (const __versa_string< _CharT, std::char_traits< _CharT >, std::allocator< _CharT >, _Base > &__lhs, const __versa_string< _CharT, std::char_traits< _CharT >, std::allocator< _CharT >, _Base > &__rhs)`
- `template<typename _Tp >`
`bool operator== (const malloc_allocator< _Tp > &, const malloc_allocator< _Tp > &)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator> (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _Tp >`
`bool operator> (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`bool operator> (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator> (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Iterator, typename _Container >`
`bool operator> (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`

- bool **operator**<> (const `__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `_Base` > &__lhs, const `__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `_Base` > &__rhs)
- template<class `_CharT`, class `_Alloc` >
bool **operator**<> (const `_Rope_iterator`< `_CharT`, `_Alloc` > &__x, const `_Rope_iterator`< `_CharT`, `_Alloc` > &__y)
 - template<class `_CharT`, class `_Alloc` >
bool **operator**<> (const `_Rope_const_iterator`< `_CharT`, `_Alloc` > &__x, const `_Rope_const_iterator`< `_CharT`, `_Alloc` > &__y)
 - template<typename `_Tp1`, typename `_Tp2` >
bool **operator**<> (const `_Pointer_adapter`< `_Tp1` > &__lhs, `_Tp2` __rhs)
 - template<class `_CharT`, class `_Alloc` >
bool **operator**<> (const `rope`< `_CharT`, `_Alloc` > &__x, const `rope`< `_CharT`, `_Alloc` > &__y)
 - template<class `_Tp`, class `_Alloc` >
bool **operator**<> (const `slist`< `_Tp`, `_Alloc` > &__SL1, const `slist`< `_Tp`, `_Alloc` > &__SL2)
 - template<typename `_Tp1`, typename `_Tp2` >
bool **operator**<> (const `_Pointer_adapter`< `_Tp1` > &__lhs, const `_Pointer_adapter`< `_Tp2` > &__rhs)
 - template<typename `_CharT`, typename `_Traits`, typename `_Alloc`, template< typename, typename, typename > class `_Base`>
bool **operator**<> (const `__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `_Base` > &__lhs, const `_CharT` * __rhs)
 - template<typename `_Tp` >
bool **operator**>= (const `_Pointer_adapter`< `_Tp` > &__lhs, const `_Pointer_adapter`< `_Tp` > &__rhs)
 - template<typename `_CharT`, typename `_Traits`, typename `_Alloc`, template< typename, typename, typename > class `_Base`>
bool **operator**>= (const `__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `_Base` > &__lhs, const `_CharT` * __rhs)
 - template<typename `_CharT`, typename `_Traits`, typename `_Alloc`, template< typename, typename, typename > class `_Base`>
bool **operator**>= (const `__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `_Base` > &__lhs, const `__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `_Base` > &__rhs)
 - template<typename `_Iterator`, typename `_Container` >
bool **operator**>= (const `__normal_iterator`< `_Iterator`, `_Container` > &__lhs, const `__normal_iterator`< `_Iterator`, `_Container` > &__rhs)
 - template<typename `_IteratorL`, typename `_IteratorR`, typename `_Container` >
bool **operator**>= (const `__normal_iterator`< `_IteratorL`, `_Container` > &__lhs, const `__normal_iterator`< `_IteratorR`, `_Container` > &__rhs)
 - template<typename `_Tp1`, typename `_Tp2` >
bool **operator**>= (const `_Pointer_adapter`< `_Tp1` > &__lhs, const `_Pointer_adapter`< `_Tp2` > &__rhs)
 - template<typename `_CharT`, typename `_Traits`, typename `_Alloc`, template< typename, typename, typename > class `_Base`>

- ```
bool operator>= (const _CharT *__lhs, const __versa_string< _CharT, _Traits,
_Alloc, _Base > &__rhs)
```
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator>= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
  - `template<class _CharT, class _Alloc >`  
`bool operator>= (const _Rope_iterator< _CharT, _Alloc > &__x, const _-`  
`Rope_iterator< _CharT, _Alloc > &__y)`
  - `template<typename _Tp1, typename _Tp2 >`  
`bool operator>= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
  - `template<class _CharT, class _Alloc >`  
`bool operator>= (const _Rope_const_iterator< _CharT, _Alloc > &__x, const`  
`_Rope_const_iterator< _CharT, _Alloc > &__y)`
  - `template<class _Tp, class _Alloc >`  
`bool operator>= (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc`  
`> &_SL2)`
  - `template<class _CharT, class _Alloc >`  
`bool operator>= (const rope< _CharT, _Alloc > &__x, const rope< _CharT,`  
`_Alloc > &__y)`
  - `template<typename _Tp, typename _Integer, typename _MonoidOperation >`  
`_Tp power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`
  - `template<typename _Tp, typename _Integer >`  
`_Tp power (_Tp __x, _Integer __n)`
  - `template<typename _InputIterator, typename _RandomAccessIterator >`  
`_RandomAccessIterator random_sample (_InputIterator __first, _InputIterator`  
`__last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last)`
  - `template<typename _InputIterator, typename _RandomAccessIterator, typename _-`  
`RandomNumberGenerator >`  
`_RandomAccessIterator random_sample (_InputIterator __first, _InputIterator`  
`__last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last,`  
`_RandomNumberGenerator &__rand)`
  - `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance, typename`  
`_RandomNumberGenerator >`  
`_OutputIterator random_sample_n (_ForwardIterator __first, _ForwardIterator`  
`__last, _OutputIterator __out, const _Distance __n, _RandomNumberGenerator`  
`&__rand)`
  - `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance >`  
`_OutputIterator random_sample_n (_ForwardIterator __first, _ForwardIterator`  
`__last, _OutputIterator __out, const _Distance __n)`
  - `void rotate (_Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)>`  
`__first, _Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __-`  
`middle, _Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __-`  
`last)`
  - `double stod (const __vstring &__str, std::size_t * __idx=0)`
  - `float stof (const __vstring &__str, std::size_t * __idx=0)`
  - `int stoi (const __vstring &__str, std::size_t * __idx=0, int __base=10)`

- long **stol** (const `__vstring` &\_\_str, std::size\_t \*\_\_idx=0, int \_\_base=10)
- long double **stold** (const `__vstring` &\_\_str, std::size\_t \*\_\_idx=0)
- long long **stoll** (const `__vstring` &\_\_str, std::size\_t \*\_\_idx=0, int \_\_base=10)
- unsigned long **stoul** (const `__vstring` &\_\_str, std::size\_t \*\_\_idx=0, int \_\_base=10)
- unsigned long long **stoull** (const `__vstring` &\_\_str, std::size\_t \*\_\_idx, int \_\_base=10)
- template<class \_Key, class \_Tp, class \_HashFn, class \_EqKey, class \_Alloc >  
void **swap** (`hash_map`< \_Key, \_Tp, \_HashFn, \_EqKey, \_Alloc > &\_\_hm1, `hash_map`< \_Key, \_Tp, \_HashFn, \_EqKey, \_Alloc > &\_\_hm2)
- template<typename \_CharT, typename \_Traits, typename \_Alloc, template< typename, typename, typename > class \_Base>  
void **swap** (`__versa_string`< \_CharT, \_Traits, \_Alloc, \_Base > &\_\_lhs, `__versa_string`< \_CharT, \_Traits, \_Alloc, \_Base > &\_\_rhs)
- template<class \_Val, class \_Key, class \_HF, class \_Extract, class \_EqKey, class \_All >  
void **swap** (hashtable< \_Val, \_Key, \_HF, \_Extract, \_EqKey, \_All > &\_\_ht1, hashtable< \_Val, \_Key, \_HF, \_Extract, \_EqKey, \_All > &\_\_ht2)
- template<class \_CharT, class \_\_Alloc >  
void **swap** (\_Rope\_char\_ref\_proxy< \_CharT, \_\_Alloc > \_\_a, \_Rope\_char\_ref\_proxy< \_CharT, \_\_Alloc > \_\_b)
- template<class \_Key, class \_Tp, class \_HashFn, class \_EqKey, class \_Alloc >  
void **swap** (`hash_multimap`< \_Key, \_Tp, \_HashFn, \_EqKey, \_Alloc > &\_\_hm1, `hash_multimap`< \_Key, \_Tp, \_HashFn, \_EqKey, \_Alloc > &\_\_hm2)
- template<class \_Val, class \_HashFcn, class \_EqualKey, class \_Alloc >  
void **swap** (`hash_multiset`< \_Val, \_HashFcn, \_EqualKey, \_Alloc > &\_\_hs1, `hash_multiset`< \_Val, \_HashFcn, \_EqualKey, \_Alloc > &\_\_hs2)
- template<typename \_Tp >  
void **swap** (`__ExtPtr_allocator`< \_Tp > &\_\_larg, `__ExtPtr_allocator`< \_Tp > &\_\_rarg)
- template<typename \_Cond >  
void **swap** (`throw_value_base`< \_Cond > &\_\_a, `throw_value_base`< \_Cond > &\_\_b)
- template<class \_Val, class \_HashFcn, class \_EqualKey, class \_Alloc >  
void **swap** (`hash_set`< \_Val, \_HashFcn, \_EqualKey, \_Alloc > &\_\_hs1, `hash_set`< \_Val, \_HashFcn, \_EqualKey, \_Alloc > &\_\_hs2)
- template<class \_Tp, class \_Alloc >  
void **swap** (`slist`< \_Tp, \_Alloc > &\_\_x, `slist`< \_Tp, \_Alloc > &\_\_y)
- template<class \_CharT, class \_Alloc >  
void **swap** (`rope`< \_CharT, \_Alloc > &\_\_x, `rope`< \_CharT, \_Alloc > &\_\_y)
- `__vstring to_string` (unsigned long long \_\_val)
- `__vstring to_string` (double \_\_val)
- `__vstring to_string` (long \_\_val)
- `__vstring to_string` (unsigned long \_\_val)
- `__vstring to_string` (long double \_\_val)

- `__vstring to_string` (int \_\_val)
- `__vstring to_string` (unsigned \_\_val)
- `__vstring to_string` (long long \_\_val)
- `__vstring to_string` (float \_\_val)
- `__wvstring to_wstring` (float \_\_val)
- `__wvstring to_wstring` (long double \_\_val)
- `__wvstring to_wstring` (unsigned \_\_val)
- `__wvstring to_wstring` (unsigned long \_\_val)
- `__wvstring to_wstring` (double \_\_val)
- `__wvstring to_wstring` (long \_\_val)
- `__wvstring to_wstring` (int \_\_val)
- `__wvstring to_wstring` (long long \_\_val)
- `__wvstring to_wstring` (unsigned long long \_\_val)
- `template<typename _InputIter, typename _Size, typename _ForwardIter >`  
`pair< _InputIter, _ForwardIter > uninitialized_copy_n` ( \_InputIter \_\_first, \_Size  
\_\_count, \_ForwardIter \_\_result)

## Variables

- static const `_Lock_policy` `__default_lock_policy`
- static const unsigned long `__stl_prime_list` [`_S_num_primes`]
- static `_Atomic_word` int `__val` `__val`
- `rope< _CharT, _Alloc >` `identity_element` ( `_Rope_Concat_fn< _CharT, _-`  
`Alloc >`)

### 4.1.1 Detailed Description

GNU extensions for public use.

### 4.1.2 Function Documentation

#### 4.1.2.1 `template<typename _ToType, typename _FromType > _ToType` `__gnu_cxx::__static_pointer_cast` ( const `_FromType` & `__arg` ) `[inline]`

Casting operations for cases where `_FromType` is not a standard pointer. `_ToType` can be a standard or non-standard pointer. Given that `_FromType` is not a pointer, it must have a `get()` method that returns the standard pointer equivalent of the address it points to, and must have an `element_type` typedef which names the type it points to.

Definition at line 68 of file `cast.h`.

**4.1.2.2** `template<typename _ToType , typename _FromType > _ToType  
__gnu_cxx::__static_pointer_cast ( _FromType * __arg ) [inline]`

Casting operations for cases where \_FromType is a standard pointer. \_ToType can be a standard or non-standard pointer.

Definition at line 96 of file cast.h.

**4.1.2.3** `size_t __gnu_cxx::_Bit_scan_forward ( size_t __num ) [inline]`

Generic Version of the bsf instruction.

Definition at line 517 of file bitmap\_allocator.h.

Referenced by \_\_gnu\_cxx::bitmap\_allocator< \_Tp >::\_M\_allocate\_single\_object().

**4.1.2.4** `template<typename _CharT , typename _Traits , typename _Alloc ,  
template< typename, typename, typename > class _Base> bool  
__gnu_cxx::operator!=( const __versa_string< _CharT, _Traits,  
_Alloc, _Base > & __lhs, const _CharT * __rhs ) [inline]`

Test difference of string and C string.

#### Parameters

*\_\_lhs* String.

*\_\_rhs* C string.

#### Returns

True if *\_\_lhs.compare(\_\_rhs) != 0*. False otherwise.

Definition at line 2247 of file vstring.h.

**4.1.2.5** `template<typename _CharT , typename _Traits , typename _Alloc ,  
template< typename, typename, typename > class _Base> bool  
__gnu_cxx::operator!=( const __versa_string< _CharT, _Traits,  
_Alloc, _Base > & __lhs, const __versa_string< _CharT, _Traits,  
_Alloc, _Base > & __rhs ) [inline]`

Test difference of two strings.

**Parameters**

`__lhs` First string.  
`__rhs` Second string.

**Returns**

True if `__lhs.compare(__rhs) != 0`. False otherwise.

Definition at line 2221 of file `vstring.h`.

```
4.1.2.6 template<typename _CharT , typename _Traits , typename _Alloc ,
template< typename, typename, typename > class _Base> bool
__gnu_cxx::operator!=(const _CharT * __lhs, const __versa_string<
_CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Test difference of C string and string.

**Parameters**

`__lhs` C string.  
`__rhs` String.

**Returns**

True if `__rhs.compare(__lhs) != 0`. False otherwise.

Definition at line 2234 of file `vstring.h`.

```
4.1.2.7 template<typename _CharT , typename _Traits , typename
_Alloc , template< typename, typename, typename > class
_Base> __versa_string< _CharT, _Traits, _Alloc, _Base >
__gnu_cxx::operator+ (const __versa_string< _CharT, _Traits, _Alloc,
_Base > & __lhs, const __versa_string< _CharT, _Traits, _Alloc,
_Base > & __rhs)
```

Concatenate two strings.

**Parameters**

`__lhs` First string.  
`__rhs` Last string.



**Returns**

New string with value of `__lhs` followed by `__rhs`.

Definition at line 181 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::reserve()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**4.1.2.8** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string<_CharT, _Traits, _Alloc, _Base> __gnu_cxx::operator+( const _CharT * __lhs, const __versa_string<_CharT, _Traits, _Alloc, _Base> & __rhs )`

Concatenate C string and string.

**Parameters**

`__lhs` First string.

`__rhs` Last string.

**Returns**

New string with value of `__lhs` followed by `__rhs`.

Definition at line 194 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**4.1.2.9** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string<_CharT, _Traits, _Alloc, _Base> __gnu_cxx::operator+( _CharT __lhs, const __versa_string<_CharT, _Traits, _Alloc, _Base> & __rhs )`

Concatenate character and string.

**Parameters**

`__lhs` First string.

`__rhs` Last string.

**Returns**

New string with `__lhs` followed by `__rhs`.

Definition at line 211 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::push_back()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::reserve()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**4.1.2.10** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ ( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const _CharT * __rhs )`

Concatenate string and C string.

**Parameters**

`__lhs` First string.

`__rhs` Last string.

**Returns**

New string with `__lhs` followed by `__rhs`.

Definition at line 224 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**4.1.2.11** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ ( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, _CharT __rhs )`

Concatenate string and character.

**Parameters**

`__lhs` First string.

`__rhs` Last string.

### Returns

New string with `__lhs` followed by `__rhs`.

Definition at line 241 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::push_back()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::reserve()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**4.1.2.12** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator< ( const __versa_string<_CharT, _Traits, _Alloc, _Base> & __lhs, const _CharT* __rhs ) [inline]`

Test if string precedes C string.

### Parameters

`__lhs` String.

`__rhs` C string.

### Returns

True if `__lhs` precedes `__rhs`. False otherwise.

Definition at line 2274 of file `vstring.h`.

**4.1.2.13** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator< ( const _CharT* __lhs, const __versa_string<_CharT, _Traits, _Alloc, _Base> & __rhs ) [inline]`

Test if C string precedes string.

### Parameters

`__lhs` C string.

`__rhs` String.

**Returns**

True if `__lhs` precedes `__rhs`. False otherwise.

Definition at line 2287 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base>::compare()`.

**4.1.2.14** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator< ( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]`

Test if string precedes string.

**Parameters**

`__lhs` First string.  
`__rhs` Second string.

**Returns**

True if `__lhs` precedes `__rhs`. False otherwise.

Definition at line 2261 of file `vstring.h`.

**4.1.2.15** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator<= ( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const _CharT * __rhs ) [inline]`

Test if string doesn't follow C string.

**Parameters**

`__lhs` String.  
`__rhs` C string.

**Returns**

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 2354 of file `vstring.h`.

**4.1.2.16** `template<typename _CharT, typename _Traits, typename _Alloc  
, template< typename, typename, typename > class _Base>  
bool __gnu_cxx::operator<= ( const _CharT * __lhs, const  
__versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs )  
[inline]`

Test if C string doesn't follow string.

#### Parameters

`__lhs` C string.

`__rhs` String.

#### Returns

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 2367 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base  
>::compare()`.

**4.1.2.17** `template<typename _CharT, typename _Traits, typename _Alloc  
, template< typename, typename, typename > class _Base> bool  
__gnu_cxx::operator<= ( const __versa_string< _CharT, _Traits,  
_Alloc, _Base > & __lhs, const __versa_string< _CharT, _Traits,  
_Alloc, _Base > & __rhs ) [inline]`

Test if string doesn't follow string.

#### Parameters

`__lhs` First string.

`__rhs` Second string.

#### Returns

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 2341 of file `vstring.h`.

**4.1.2.18** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool  
__gnu_cxx::operator==( const __versa_string< _CharT, _Traits,  
_Alloc, _Base > & __lhs, const __versa_string< _CharT, _Traits,  
_Alloc, _Base > & __rhs ) [inline]`

Test equivalence of two strings.

#### Parameters

`__lhs` First string.  
`__rhs` Second string.

#### Returns

True if `__lhs.compare(__rhs) == 0`. False otherwise.

Definition at line 2170 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

**4.1.2.19** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool  
__gnu_cxx::operator==( const _CharT * __lhs, const  
__versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs )  
[inline]`

Test equivalence of C string and string.

#### Parameters

`__lhs` C string.  
`__rhs` String.

#### Returns

True if `__rhs.compare(__lhs) == 0`. False otherwise.

Definition at line 2194 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

**4.1.2.20** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool  
__gnu_cxx::operator==( const __versa_string< _CharT, _Traits,  
_Alloc, _Base > & __lhs, const _CharT* __rhs ) [inline]`

Test equivalence of string and C string.

#### Parameters

`__lhs` String.

`__rhs` C string.

#### Returns

True if `__lhs.compare(__rhs) == 0`. False otherwise.

Definition at line 2207 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

**4.1.2.21** `template<typename _Tp > bool __gnu_cxx::operator==( const  
_Pointer_adapter< _Tp > & __lhs, const _Pointer_adapter< _Tp >  
& __rhs ) [inline]`

Comparison operators for `_Pointer_adapter` defer to the base class's comparison operators, when possible.

Definition at line 533 of file `pointer.h`.

**4.1.2.22** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool  
__gnu_cxx::operator>( const _CharT* __lhs, const __versa_string<  
_CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]`

Test if C string follows string.

#### Parameters

`__lhs` C string.

`__rhs` String.

**Returns**

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 2327 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base>::compare()`.

**4.1.2.23** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator> ( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]`

Test if string follows string.

**Parameters**

`__lhs` First string.

`__rhs` Second string.

**Returns**

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 2301 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base>::compare()`.

**4.1.2.24** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator> ( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const _CharT* __rhs ) [inline]`

Test if string follows C string.

**Parameters**

`__lhs` String.

`__rhs` C string.



**Returns**

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 2314 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base>::compare()`.

**4.1.2.25** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool  
__gnu_cxx::operator>= ( const __versa_string< _CharT, _Traits,  
_Alloc, _Base > & __lhs, const _CharT* __rhs ) [inline]`

Test if string doesn't precede C string.

**Parameters**

`__lhs` String.

`__rhs` C string.

**Returns**

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 2394 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base>::compare()`.

**4.1.2.26** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool  
__gnu_cxx::operator>= ( const __versa_string< _CharT, _Traits,  
_Alloc, _Base > & __lhs, const __versa_string< _CharT, _Traits,  
_Alloc, _Base > & __rhs ) [inline]`

Test if string doesn't precede string.

**Parameters**

`__lhs` First string.

`__rhs` Second string.

**Returns**

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 2381 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

```
4.1.2.27 template<typename _CharT , typename _Traits , typename _Alloc
, template< typename, typename, typename > class _Base>
bool __gnu_cxx::operator>= (const _CharT * __lhs, const
__versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)
[inline]
```

Test if C string doesn't precede string.

**Parameters**

`__lhs` C string.

`__rhs` String.

**Returns**

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 2407 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

```
4.1.2.28 template<typename _CharT , typename _Traits , typename _Alloc
, template< typename, typename, typename > class _Base> void
__gnu_cxx::swap (__versa_string< _CharT, _Traits, _Alloc, _Base >
& __lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs
) [inline]
```

Swap contents of two strings.

**Parameters**

`__lhs` First string.

`__rhs` Second string.

Exchanges the contents of `__lhs` and `__rhs` in constant time.

Definition at line 2421 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::swap()`.

## 4.2 `__gnu_cxx::__detail` Namespace Reference

Implementation details not part of the namespace `__gnu_cxx` interface.

### Classes

- class `__mini_vector`  
*`__mini_vector<>` is a stripped down version of the full-fledged `std::vector<>`.*
- class `__Bitmap_counter`  
*The bitmap counter which acts as the bitmap manipulator, and manages the bit-manipulation functions and the searching and identification functions on the bit-map.*
- class `__Ffit_finder`  
*The class which acts as a predicate for applying the first-fit memory allocation policy for the bitmap allocator.*

### Enumerations

- enum { `_S_max_rope_depth` }
- enum { `bits_per_byte`, `bits_per_block` }
- enum `_Tag` { `_S_leaf`, `_S_concat`, `_S_substringfn`, `_S_function` }

### Functions

- void `__bit_allocate` (size\_t \* \_\_pmap, size\_t \_\_pos) throw ()
- void `__bit_free` (size\_t \* \_\_pmap, size\_t \_\_pos) throw ()
- template<typename \_ForwardIterator, typename \_Tp, typename \_Compare>  
\_ForwardIterator `__lower_bound` (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, const \_Tp & \_\_val, \_Compare \_\_comp)
- template<typename \_AddrPair>  
size\_t `__num_bitmaps` (\_AddrPair \_\_ap)
- template<typename \_AddrPair>  
size\_t `__num_blocks` (\_AddrPair \_\_ap)

### 4.2.1 Detailed Description

Implementation details not part of the namespace `__gnu_cxx` interface.

### 4.2.2 Function Documentation

**4.2.2.1** `void __gnu_cxx::__detail::__bit_allocate ( size_t * __pbmap, size_t __pos ) throw () [inline]`

Mark a memory address as allocated by re-setting the corresponding bit in the bit-map.

Definition at line 492 of file `bitmap_allocator.h`.

Referenced by `__gnu_cxx::bitmap_allocator< _Tp >::_M_allocate_single_object()`.

**4.2.2.2** `void __gnu_cxx::__detail::__bit_free ( size_t * __pbmap, size_t __pos ) throw () [inline]`

Mark a memory address as free by setting the corresponding bit in the bit-map.

Definition at line 503 of file `bitmap_allocator.h`.

Referenced by `__gnu_cxx::bitmap_allocator< _Tp >::_M_deallocate_single_object()`.

**4.2.2.3** `template<typename _AddrPair > size_t __gnu_cxx::__detail::__num_bitmaps ( _AddrPair __ap ) [inline]`

The number of Bit-maps pointed to by the address pair passed to the function.

Definition at line 280 of file `bitmap_allocator.h`.

References `__num_blocks()`.

Referenced by `__gnu_cxx::bitmap_allocator< _Tp >::_M_allocate_single_object()`, and `__gnu_cxx::bitmap_allocator< _Tp >::_M_deallocate_single_object()`.

**4.2.2.4** `template<typename _AddrPair > size_t __gnu_cxx::__detail::__num_blocks ( _AddrPair __ap )  
[inline]`

The number of Blocks pointed to by the address pair passed to the function.

Definition at line 272 of file bitmap\_allocator.h.

Referenced by \_\_num\_bitmaps().

## 4.3 \_\_gnu\_cxx::typelist Namespace Reference

GNU typelist extensions for public compile-time use.

### Functions

- `template<typename Fn , typename Typelist >  
void apply (Fn &, Typelist)`
- `template<typename Fn , typename TypelistT , typename TypelistV >  
void apply_generator (Fn &fn, TypelistT, TypelistV)`
- `template<typename Fn , typename Typelist >  
void apply_generator (Fn &fn, Typelist)`
- `template<typename Gn , typename TypelistT , typename TypelistV >  
void apply_generator (Gn &, TypelistT, TypelistV)`
- `template<typename Gn , typename Typelist >  
void apply\_generator (Gn &, Typelist)`

### 4.3.1 Detailed Description

GNU typelist extensions for public compile-time use.

### 4.3.2 Function Documentation

**4.3.2.1** `template<typename Gn , typename Typelist > void  
__gnu_cxx::typelist::apply_generator ( Gn &, Typelist )`

Apply all typelist types to generator functor.

## 4.4 \_\_gnu\_debug Namespace Reference

GNU debug classes for public use.

### Classes

- class [\\_After\\_nth\\_from](#)
- struct [\\_BeforeBeginHelper](#)
- class [\\_Equal\\_to](#)
- class [\\_Not\\_equal\\_to](#)
- class [\\_Safe\\_iterator](#)  
*Safe iterator wrapper.*
- class [\\_Safe\\_iterator\\_base](#)  
*Basic functionality for a safe iterator.*
- class [\\_Safe\\_sequence](#)  
*Base class for constructing a safe sequence type that tracks iterators that reference it.*
- class [\\_Safe\\_sequence\\_base](#)  
*Base class that supports tracking of iterators that reference a sequence.*
- class [basic\\_string](#)  
*Class `std::basic_string` with safety/checking/debug instrumentation.*

### Typedefs

- typedef [basic\\_string](#)< char > **string**
- typedef [basic\\_string](#)< wchar\_t > **wstring**

### Enumerations

- enum **\_Debug\_msg\_id** {  
    \_\_msg\_valid\_range, \_\_msg\_insert\_singular, \_\_msg\_insert\_different, \_\_-  
    msg\_erase\_bad,  
    \_\_msg\_erase\_different, \_\_msg\_subscript\_oob, \_\_msg\_empty, \_\_msg\_-  
    unpartitioned,  
    \_\_msg\_unpartitioned\_pred, \_\_msg\_unsorted, \_\_msg\_unsorted\_pred, \_\_-  
    msg\_not\_heap,

```

__msg_not_heap_pred, __msg_bad_bitset_write, __msg_bad_bitset_read,
__msg_bad_bitset_flip,
__msg_self_splice, __msg_splice_alloc, __msg_splice_bad, __msg_splice_
other,
__msg_splice_overlap, __msg_init_singular, __msg_init_copy_singular, __
msg_init_const_singular,
__msg_copy_singular, __msg_bad_deref, __msg_bad_inc, __msg_bad_dec,
__msg_iter_subscript_oob, __msg_advance_oob, __msg_retreat_oob, __
msg_iter_compare_bad,
__msg_compare_different, __msg_iter_order_bad, __msg_order_different,
__msg_distance_bad,
__msg_distance_different, __msg_deref_istream, __msg_inc_istream, __
msg_output_ostream,
__msg_deref_istreambuf, __msg_inc_istreambuf, __msg_insert_after_end,
__msg_erase_after_bad,
__msg_valid_range2 }

```

## Functions

- `template<typename _Iterator >`  
`_Siter_base< _Iterator >::iterator_type __base ( _Iterator __it)`
- `template<typename _Iterator >`  
`bool __check_dereferenceable ( _Iterator &)`
- `template<typename _Tp >`  
`bool __check_dereferenceable (const _Tp * __ptr)`
- `template<typename _Iterator, typename _Sequence >`  
`bool __check_dereferenceable (const _Safe_iterator< _Iterator, _Sequence >`  
`& __x)`
- `template<typename _ForwardIterator, typename _Tp >`  
`bool __check_partitioned_lower ( _ForwardIterator __first, _ForwardIterator _`  
`__last, const _Tp & __value)`
- `template<typename _ForwardIterator, typename _Tp, typename _Pred >`  
`bool __check_partitioned_lower ( _ForwardIterator __first, _ForwardIterator _`  
`__last, const _Tp & __value, _Pred __pred)`
- `template<typename _ForwardIterator, typename _Tp >`  
`bool __check_partitioned_upper ( _ForwardIterator __first, _ForwardIterator`  
`__last, const _Tp & __value)`
- `template<typename _ForwardIterator, typename _Tp, typename _Pred >`  
`bool __check_partitioned_upper ( _ForwardIterator __first, _ForwardIterator`  
`__last, const _Tp & __value, _Pred __pred)`
- `template<typename _Iterator >`  
`bool __check_singular ( _Iterator &)`

- `template<typename _Iterator, typename _Sequence >`  
`bool __check_singular (const _Safe_iterator< _Iterator, _Sequence > &__x)`
- `template<typename _Tp >`  
`bool __check_singular (const _Tp * __ptr)`
- `bool __check_singular_aux (const void *)`
- `bool __check_singular_aux (const _Safe_iterator_base * __x)`
- `template<typename _InputIterator >`  
`bool __check_sorted (const _InputIterator &__first, const _InputIterator &__last)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool __check_sorted (const _InputIterator &__first, const _InputIterator &__last, _Predicate __pred)`
- `template<typename _InputIterator >`  
`bool __check_sorted_aux (const _InputIterator &, const _InputIterator &, std::input_iterator_tag)`
- `template<typename _ForwardIterator >`  
`bool __check_sorted_aux (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool __check_sorted_aux (const _InputIterator &, const _InputIterator &, _Predicate, std::input_iterator_tag)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`bool __check_sorted_aux (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, std::forward_iterator_tag)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`bool __check_sorted_set (const _InputIterator1 &__first, const _InputIterator1 &__last, const _InputIterator2 &)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Predicate >`  
`bool __check_sorted_set (const _InputIterator1 &__first, const _InputIterator1 &__last, const _InputIterator2 &, _Predicate __pred)`
- `template<typename _InputIterator >`  
`bool __check_sorted_set_aux (const _InputIterator &__first, const _InputIterator &__last, std::__true_type)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool __check_sorted_set_aux (const _InputIterator &__first, const _InputIterator &__last, _Predicate __pred, std::__true_type)`
- `template<typename _InputIterator >`  
`bool __check_sorted_set_aux (const _InputIterator &, const _InputIterator &, std::__false_type)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool __check_sorted_set_aux (const _InputIterator &, const _InputIterator &, _Predicate, std::__false_type)`
- `template<typename _CharT >`  
`const _CharT * __check_string (const _CharT * __s)`



- `template<typename _CharT, typename _Integer >`  
`const _CharT * \_\_check\_string (const _CharT *__s, const _Integer &__n __-`  
`attribute__((__unused__)))`
- `template<typename _InputIterator >`  
`_InputIterator \_\_check\_valid\_range (const _InputIterator &__first, const _`  
`InputIterator &__last __attribute__((__unused__)))`
- `template<typename _InputIterator >`  
`bool \_\_valid\_range (const _InputIterator &__first, const _InputIterator &__last)`
- `template<typename _Iterator, typename _Sequence >`  
`bool \_\_valid\_range (const \_Safe\_iterator< _Iterator, _Sequence > &__first,`  
`const \_Safe\_iterator< _Iterator, _Sequence > &__last)`
- `template<typename _Integral >`  
`bool \_\_valid\_range\_aux (const _Integral &, const _Integral &, std::__true_type)`
- `template<typename _InputIterator >`  
`bool \_\_valid\_range\_aux (const _InputIterator &__first, const _InputIterator &_`  
`__last, std::__false_type)`
- `template<typename _InputIterator >`  
`bool \_\_valid\_range\_aux2 (const _InputIterator &, const _InputIterator &,`  
`std::input\_iterator\_tag)`
- `template<typename _RandomAccessIterator >`  
`bool \_\_valid\_range\_aux2 (const _RandomAccessIterator &__first, const _`  
`RandomAccessIterator &__last, std::random\_access\_iterator\_tag)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`std::basic\_istream< _CharT, _Traits > & getline (std::basic\_istream< _CharT,`  
`_Traits > &__is, basic\_string< _CharT, _Traits, _Allocator > &__str, _CharT`  
`__delim)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`std::basic\_istream< _CharT, _Traits > & getline (std::basic\_istream< _CharT,`  
`_Traits > &__is, basic\_string< _CharT, _Traits, _Allocator > &__str)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator!= (const \_Safe\_iterator< _IteratorL, _Sequence > &__lhs, const`  
`\_Safe\_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator!= (const \_Safe\_iterator< _Iterator, _Sequence > &__lhs, const`  
`\_Safe\_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator!= (const _CharT *__lhs, const basic\_string< _CharT, _Traits, _`  
`Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator!= (const basic\_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator!= (const basic\_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const basic\_string< _CharT, _Traits, _Allocator > &__rhs)`

- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`basic_string< _CharT, _Traits, _Allocator > operator+ (const _CharT *__lhs,`  
`const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`basic_string< _CharT, _Traits, _Allocator > operator+ (const basic_string<`  
`_CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`_Safe_iterator< _Iterator, _Sequence > operator+ (typename _Safe_iterator<`  
`_Iterator, _Sequence >::difference_type __n, const _Safe_iterator< _Iterator, _`  
`Sequence > &__i)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`basic_string< _CharT, _Traits, _Allocator > operator+ (const basic_string<`  
`_CharT, _Traits, _Allocator > &__lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`basic_string< _CharT, _Traits, _Allocator > operator+ (const basic_string<`  
`_CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _`  
`Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`basic_string< _CharT, _Traits, _Allocator > operator+ (_CharT __lhs, const`  
`basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`_Safe_iterator< _Iterator, _Sequence >::difference_type operator- (const _`  
`Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator,`  
`_Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`_Safe_iterator< _IteratorL, _Sequence >::difference_type operator- (const _`  
`Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _`  
`IteratorR, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator< (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const`  
`_Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator< (const basic_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator< (const basic_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator< (const _CharT *__lhs, const basic_string< _CharT, _Traits, _`  
`Allocator > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator< (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const`  
`_Safe_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream<`

```
_CharT, _Traits > &__os, const basic_string< _CharT, _Traits, _Allocator >
&__str)
```

- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator<= (const basic\_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator<= (const basic\_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const basic\_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator<= (const _CharT *__lhs, const basic\_string< _CharT, _Traits,`  
`_Allocator > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator<= (const \_Safe\_iterator< _Iterator, _Sequence > &__lhs, const`  
`\_Safe\_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator<= (const \_Safe\_iterator< _IteratorL, _Sequence > &__lhs, const`  
`\_Safe\_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator== (const basic\_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const basic\_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator== (const \_Safe\_iterator< _IteratorL, _Sequence > &__lhs, const`  
`\_Safe\_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator== (const basic\_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const _CharT *__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator== (const \_Safe\_iterator< _Iterator, _Sequence > &__lhs, const`  
`\_Safe\_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator== (const _CharT *__lhs, const basic\_string< _CharT, _Traits,`  
`_Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator> (const basic\_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const basic\_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator> (const \_Safe\_iterator< _IteratorL, _Sequence > &__lhs, const`  
`\_Safe\_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator> (const _CharT *__lhs, const basic\_string< _CharT, _Traits, _`  
`_Allocator > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator> (const \_Safe\_iterator< _Iterator, _Sequence > &__lhs, const`  
`\_Safe\_iterator< _Iterator, _Sequence > &__rhs)`

- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator> (const basic_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator>= (const _CharT *__lhs, const basic_string< _CharT, _Traits,`  
`_Allocator > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator>= (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const`  
`_Safe_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator>= (const basic_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator>= (const basic_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator>= (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const`  
`_Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _`  
`CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Allocator > &__str)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`void swap (basic_string< _CharT, _Traits, _Allocator > &__lhs, basic_string<`  
`_CharT, _Traits, _Allocator > &__rhs)`

#### 4.4.1 Detailed Description

GNU debug classes for public use.

#### 4.4.2 Function Documentation

##### 4.4.2.1 `template<typename _Iterator > _Siter_base<_Iterator>::iterator_type` `__gnu_debug::__base ( _Iterator __it ) [inline]`

Helper function to extract base iterator of random access safe iterator in order to reduce performance impact of debug mode. Limited to random access iterator because it is the only category for which it is possible to check for correct iterators order in the `__valid_range` function thanks to the `<` operator.

Definition at line 683 of file `safe_iterator.h`.

Referenced by `std::internal()`.

**4.4.2.2** `template<typename _Iterator > bool __gnu_debug::__check_dereferenceable ( _Iterator & )  
[inline]`

Assume that some arbitrary iterator is dereferenceable, because we can't prove that it isn't.

Definition at line 70 of file functions.h.

**4.4.2.3** `template<typename _Tp > bool __gnu_debug::__check_dereferenceable ( const _Tp * __ptr )  
[inline]`

Non-NULL pointers are dereferenceable.

Definition at line 76 of file functions.h.

**4.4.2.4** `template<typename _Iterator , typename _Sequence > bool  
__gnu_debug::__check_dereferenceable ( const _Safe_iterator<  
_Iterator, _Sequence > & __x ) [inline]`

Safe iterators know if they are singular.

Definition at line 82 of file functions.h.

References `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_dereferenceable()`.

**4.4.2.5** `template<typename _Iterator , typename _Sequence > bool  
__gnu_debug::__check_singular ( const _Safe_iterator< _Iterator,  
_Sequence > & __x ) [inline]`

Safe iterators know if they are singular.

Definition at line 63 of file functions.h.

References `__gnu_debug::_Safe_iterator_base::_M_singular()`.

**4.4.2.6** `template<typename _Tp > bool __gnu_debug::__check_singular ( const _Tp * __ptr ) [inline]`

Non-NULL pointers are nonsingular.

Definition at line 57 of file functions.h.

**4.4.2.7** `bool __gnu_debug::__check_singular_aux ( const _Safe_iterator_base *  
__x ) [inline]`

Iterators that derive from `_Safe_iterator_base` but that aren't `_Safe_iterators` can be determined singular or non-singular via `_Safe_iterator_base`.

Definition at line 62 of file `safe_iterator.h`.

References `__gnu_debug::_Safe_iterator_base::M_singular()`.

**4.4.2.8** `template<typename _CharT > const _CharT*  
__gnu_debug::__check_string ( const _CharT * __s ) [inline]`

Checks that `__s` is non-NULL and then returns `__s`.

Definition at line 176 of file `functions.h`.

**4.4.2.9** `template<typename _CharT , typename _Integer > const _CharT*  
__gnu_debug::__check_string ( const _CharT * __s, const _Integer  
&__n __attribute__((unused)) ) [inline]`

Checks that `__s` is non-NULL or `__n == 0`, and then returns `__s`.

Definition at line 164 of file `functions.h`.

**4.4.2.10** `template<typename _InputIterator > bool __gnu_debug::__valid_  
range ( const _InputIterator & __first, const _InputIterator & __last  
) [inline]`

Don't know what these iterators are, or if they are even iterators (we may get an integral type for `InputIterator`), so see if they are integral and pass them on to the next phase otherwise.

Definition at line 134 of file `functions.h`.

References `__valid_range_aux()`.

**4.4.2.11** `template<typename _Iterator , typename _Sequence > bool  
__gnu_debug::__valid_range ( const _Safe_iterator< _Iterator,  
_Sequence > & __first, const _Safe_iterator< _Iterator, _Sequence >  
& __last ) [inline]`

Safe iterators know how to check if they form a valid range.

Definition at line 143 of file `functions.h`.

**4.4.2.12** `template<typename _Integral > bool __gnu_debug::__valid_range_ -  
aux ( const _Integral &, const _Integral &, std::__true_type )  
[inline]`

We say that integral types for a valid range, and defer to other routines to realize what to do with integral types instead of iterators.

Definition at line 111 of file functions.h.

Referenced by \_\_valid\_range().

**4.4.2.13** `template<typename _InputIterator > bool __gnu_debug::__valid_ -  
range_aux ( const _InputIterator & __first, const _InputIterator &  
__last, std::__false_type ) [inline]`

We have iterators, so figure out what kind of iterators that are to see if we can check the range ahead of time.

Definition at line 119 of file functions.h.

References \_\_valid\_range\_aux2().

**4.4.2.14** `template<typename _InputIterator > bool __gnu_debug::__valid_ -  
range_aux2 ( const _InputIterator &, const _InputIterator & ,  
std::input_iterator_tag ) [inline]`

Can't test for a valid range with input iterators, because iteration may be destructive. So we just assume that the range is valid.

Definition at line 101 of file functions.h.

**4.4.2.15** `template<typename _RandomAccessIterator > bool  
__gnu_debug::__valid_range_aux2 ( const _RandomAccessIterator  
& __first, const _RandomAccessIterator & __last,  
std::random_access_iterator_tag ) [inline]`

If the distance between two random access iterators is nonnegative, assume the range is valid.

Definition at line 90 of file functions.h.

Referenced by \_\_valid\_range\_aux().

## 4.5 `__gnu_internal` Namespace Reference

GNU implementation details, not for public use or export. Used only when anonymous namespaces cannot be substituted.

### 4.5.1 Detailed Description

GNU implementation details, not for public use or export. Used only when anonymous namespaces cannot be substituted.

## 4.6 `__gnu_parallel` Namespace Reference

GNU parallel code for public use.

### Classes

- struct `__accumulate_binop_reduct`  
*General reduction, using a binary operator.*
- struct `__accumulate_selector`  
*`std::accumulate()` selector.*
- struct `__adjacent_difference_selector`  
*Selector that returns the difference between two adjacent `__elements`.*
- struct `__adjacent_find_selector`  
*Test predicate on two adjacent elements.*
- class `__binder1st`  
*Similar to `std::binder1st`, but giving the argument types explicitly.*
- class `__binder2nd`  
*Similar to `std::binder2nd`, but giving the argument types explicitly.*
- struct `__count_if_selector`  
*`std::count_if()` selector.*
- struct `__count_selector`  
*`std::count()` selector.*
- struct `__fill_selector`



*std::fill() selector.*

- struct `__find_first_of_selector`  
*Test predicate on several elements.*
- struct `__find_if_selector`  
*Test predicate on a single element, used for std::find() and std::find\_if().*
- struct `__for_each_selector`  
*std::for\_each() selector.*
- struct `__generate_selector`  
*std::generate() selector.*
- struct `__generic_find_selector`  
*Base class of all \_\_gnu\_parallel::\_\_find\_template selectors.*
- struct `__generic_for_each_selector`  
*Generic \_\_selector for embarrassingly parallel functions.*
- struct `__identity_selector`  
*Selector that just returns the passed iterator.*
- struct `__inner_product_selector`  
*std::inner\_product() selector.*
- struct `__max_element_reduct`  
*Reduction for finding the maximum element, using a comparator.*
- struct `__min_element_reduct`  
*Reduction for finding the maximum element, using a comparator.*
- struct `__mismatch_selector`  
*Test inverted predicate on a single element.*
- struct `__multiway_merge_3_variant_sentinel_switch`  
*Switch for 3-way merging with \_\_sentinels turned off.*
- struct `__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`  
*Switch for 3-way merging with \_\_sentinels turned on.*
- struct `__multiway_merge_4_variant_sentinel_switch`

*Switch for 4-way merging with `__sentinels` turned off.*

- struct `__multiway_merge_4_variant_sentinel_switch`< `true`, `_RAIterIterator`, `_RAIter3`, `_DifferenceTp`, `_Compare` >

*Switch for 4-way merging with `__sentinels` turned on.*

- struct `__multiway_merge_k_variant_sentinel_switch`

*Switch for k-way merging with `__sentinels` turned on.*

- struct `__multiway_merge_k_variant_sentinel_switch`< `false`, `__stable`, `_RAIterIterator`, `_RAIter3`, `_DifferenceTp`, `_Compare` >

*Switch for k-way merging with `__sentinels` turned off.*

- struct `__replace_if_selector`

*`std::replace()` selector.*

- struct `__replace_selector`

*`std::replace()` selector.*

- struct `__transform1_selector`

*`std::transform()` `__selector`; one input sequence variant.*

- struct `__transform2_selector`

*`std::transform()` `__selector`; two input sequences variant.*

- class `__unary_negate`

*Similar to `std::unary_negate`, but giving the argument types explicitly.*

- struct `_DRandomShufflingGlobalData`

*Data known to every thread participating in `__gnu_parallel::__parallel_random_shuffle()`.*

- struct `_DRSSorterPU`

*Local data for a thread participating in `__gnu_parallel::__parallel_random_shuffle()`.*

- struct `_DummyReduct`

*Reduction function doing nothing.*

- class `_EqualFromLess`

*Constructs predicate for equality from strict weak ordering predicate.*

- struct `_EqualTo`

Similar to `std::equal_to`, but allows two different types.

- class `_GuardedIterator`  
*\_Iterator wrapper supporting an implicit supremum at the end of the sequence, dominating all comparisons.*
- class `_IteratorPair`  
*A pair of iterators. The usual iterator operations are applied to both child iterators.*
- class `_IteratorTriple`  
*A triple of iterators. The usual iterator operations are applied to all three child iterators.*
- struct `_Job`  
*One \_\_job for a certain thread.*
- struct `_Less`  
*Similar to `std::less`, but allows two different types.*
- class `_Lexicographic`  
*Compare \_\_a pair of types lexicographically, ascending.*
- class `_LexicographicReverse`  
*Compare \_\_a pair of types lexicographically, descending.*
- class `_LoserTree`  
*Stable `_LoserTree` variant.*
- class `_LoserTree< false, _Tp, _Compare >`  
*Unstable `_LoserTree` variant.*
- class `_LoserTreeBase`  
*Guarded loser/tournament tree.*
- class `_LoserTreePointer`  
*Stable `_LoserTree` implementation.*
- class `_LoserTreePointer< false, _Tp, _Compare >`  
*Unstable `_LoserTree` implementation.*
- class `_LoserTreePointerBase`  
*Base class of `_LoserTree` implementation using pointers.*

- class `_LoserTreePointerUnguarded`  
*Stable unguarded `_LoserTree` variant storing pointers.*
- class `_LoserTreePointerUnguarded< false, _Tp, _Compare >`  
*Unstable unguarded `_LoserTree` variant storing pointers.*
- class `_LoserTreePointerUnguardedBase`  
*Unguarded loser tree, keeping only pointers to the elements in the tree structure.*
- struct `_LoserTreeTraits`  
*Traits for determining whether the loser tree should use pointers or copies.*
- class `_LoserTreeUnguarded`  
*Stable implementation of unguarded `_LoserTree`.*
- class `_LoserTreeUnguarded< false, _Tp, _Compare >`  
*Non-Stable implementation of unguarded `_LoserTree`.*
- class `_LoserTreeUnguardedBase`  
*Base class for unguarded `_LoserTree` implementation.*
- struct `_Multiplies`  
*Similar to `std::multiplies`, but allows two different types.*
- struct `_Nothing`  
*Functor doing nothing.*
- struct `_Piece`  
*Subsequence description.*
- struct `_Plus`  
*Similar to `std::plus`, but allows two different types.*
- struct `_PMWMSortingData`  
*Data accessed by all threads.*
- class `_PseudoSequence`  
*Sequence that conceptually consists of multiple copies of the same element. The copies are not stored explicitly, of course.*
- class `_PseudoSequenceIterator`  
*Iterator associated with `__gnu_parallel::_PseudoSequence`. If features the usual random-access iterator functionality.*

- struct [\\_QSBThreadLocal](#)  
*Information local to one thread in the parallel quicksort run.*
- class [\\_RandomNumber](#)  
*Random number generator, based on the Mersenne twister.*
- class [\\_RestrictedBoundedConcurrentQueue](#)  
*Double-ended queue of bounded size, allowing lock-free atomic access. [push\\_front\(\)](#) and [pop\\_front\(\)](#) must not be called concurrently to each other, while [pop\\_back\(\)](#) can be called concurrently at all times. [empty\(\)](#), [size\(\)](#), and [top\(\)](#) are intentionally not provided. Calling them would not make sense in a concurrent setting.*
- struct [\\_SamplingSorter](#)  
*Stable sorting functor.*
- struct [\\_SamplingSorter< false, \\_RAIter, \\_StrictWeakOrdering >](#)  
*Non-\_\_stable sorting functor.*
- struct [\\_Settings](#)  
*class [\\_Settings](#) /// Run-time settings for the parallel mode including all tunable parameters.*
- struct [\\_SplitConsistently](#)  
*Split consistently.*
- struct [\\_SplitConsistently< false, \\_RAIter, \\_Compare, \\_SortingPlacesIterator >](#)  
*Split by sampling.*
- struct [\\_SplitConsistently< true, \\_RAIter, \\_Compare, \\_SortingPlacesIterator >](#)  
*Split by exact splitting.*
- struct [balanced\\_quicksort\\_tag](#)  
*Forces parallel sorting using balanced quicksort at compile time.*
- struct [balanced\\_tag](#)  
*Recommends parallel execution using dynamic load-balancing at compile time.*
- struct [constant\\_size\\_blocks\\_tag](#)  
*Selects the constant block size variant for `std::find()`.*
- struct [default\\_parallel\\_tag](#)

*Recommends parallel execution using the default parallel algorithm.*

- struct [equal\\_split\\_tag](#)  
*Selects the equal splitting variant for `std::find()`.*
- struct [exact\\_tag](#)  
*Forces parallel merging with exact splitting, at compile time.*
- struct [find\\_tag](#)  
*Base class for `std::find()` variants.*
- struct [growing\\_blocks\\_tag](#)  
*Selects the growing block size variant for `std::find()`.*
- struct [multiway\\_mergesort\\_exact\\_tag](#)  
*Forces parallel sorting using multiway mergesort with exact splitting at compile time.*
- struct [multiway\\_mergesort\\_sampling\\_tag](#)  
*Forces parallel sorting using multiway mergesort with splitting by sampling at compile time.*
- struct [multiway\\_mergesort\\_tag](#)  
*Forces parallel sorting using multiway mergesort at compile time.*
- struct [omp\\_loop\\_static\\_tag](#)  
*Recommends parallel execution using OpenMP static load-balancing at compile time.*
- struct [omp\\_loop\\_tag](#)  
*Recommends parallel execution using OpenMP dynamic load-balancing at compile time.*
- struct [parallel\\_tag](#)  
*Recommends parallel execution at compile time, optionally using a user-specified number of threads.*
- struct [quicksort\\_tag](#)  
*Forces parallel sorting using unbalanced quicksort at compile time.*
- struct [sampling\\_tag](#)  
*Forces parallel merging with exact splitting, at compile time.*
- struct [sequential\\_tag](#)  
*Forces sequential execution at compile time.*

- struct [unbalanced\\_tag](#)

*Recommends parallel execution using static load-balancing at compile time.*

## Typedefs

- typedef unsigned short [\\_BinIndex](#)
- typedef int64\_t [\\_CASable](#)
- typedef uint64\_t [\\_SequenceIndex](#)
- typedef uint16\_t [\\_ThreadIndex](#)

## Enumerations

- enum [\\_AlgorithmStrategy](#) { **heuristic**, **force\_sequential**, **force\_parallel** }
- enum [\\_FindAlgorithm](#) { **GROWING\_BLOCKS**, **CONSTANT\_SIZE\_BLOCKS**, **EQUAL\_SPLIT** }
- enum [\\_MultiwayMergeAlgorithm](#) { **LOSER\_TREE** }
- enum [\\_Parallelism](#) {   
 [sequential](#), [parallel\\_unbalanced](#), [parallel\\_balanced](#), [parallel\\_omp\\_loop](#),   
 [parallel\\_omp\\_loop\\_static](#), [parallel\\_taskqueue](#) }
- enum [\\_PartialSumAlgorithm](#) { **RECURSIVE**, **LINEAR** }
- enum [\\_SortAlgorithm](#) { **MWMS**, **QS**, **QS\_BALANCED** }
- enum [\\_SplittingAlgorithm](#) { **SAMPLING**, **EXACT** }

## Functions

- template<typename \_RAIter, typename \_DifferenceTp >  
void [\\_\\_calc\\_borders](#) (\_RAIter \_\_elements, \_DifferenceTp \_\_length, \_DifferenceTp \*\_\_off)
- template<typename \_Tp >  
bool [\\_\\_compare\\_and\\_swap](#) (volatile \_Tp \*\_\_ptr, \_Tp \_\_comparand, \_Tp \_\_replacement)
- bool [\\_\\_compare\\_and\\_swap\\_32](#) (volatile int32\_t \*\_\_ptr, int32\_t \_\_comparand, int32\_t \_\_replacement)
- bool [\\_\\_compare\\_and\\_swap\\_64](#) (volatile int64\_t \*\_\_ptr, int64\_t \_\_comparand, int64\_t \_\_replacement)
- template<typename \_IIter, typename \_OutputIterator >  
\_OutputIterator [\\_\\_copy\\_tail](#) (std::pair< \_IIter, \_IIter > \_\_b, std::pair< \_IIter, \_IIter > \_\_e, \_OutputIterator \_\_r)
- void [\\_\\_decode2](#) (\_CASable \_\_x, int &\_\_a, int &\_\_b)

- `template<typename _RAIter, typename _DifferenceTp >`  
`void __determine_samples (\_PMWSSortingData< _RAIter > *__sd, _-`  
`DifferenceTp __num_samples)`
- [\\_CASable \\_\\_encode2](#) (int \_\_a, int \_\_b)
- `template<typename _Tp >`  
`_Tp __fetch_and_add (volatile _Tp *__ptr, _Tp __addend)`
- `int32_t __fetch_and_add_32 (volatile int32_t *__ptr, int32_t __addend)`
- `int64_t __fetch_and_add_64 (volatile int64_t *__ptr, int64_t __addend)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >`  
`std::pair< _RAIter1, _RAIter2 > \_\_find\_template (_RAIter1 __begin1, _-`  
`RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >`  
`std::pair< _RAIter1, _RAIter2 > \_\_find\_template (_RAIter1 __begin1, _-`  
`RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector,`  
`equal\_split\_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >`  
`std::pair< _RAIter1, _RAIter2 > \_\_find\_template (_RAIter1 __begin1, _-`  
`RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector,`  
`growing\_blocks\_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >`  
`std::pair< _RAIter1, _RAIter2 > \_\_find\_template (_RAIter1 __begin1, _-`  
`RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector,`  
`constant\_size\_blocks\_tag)`
- `template<typename _IIter, typename _UserOp, typename _Functionality, typename _Red, type-`  
`name _Result >`  
`_UserOp \_\_for\_each\_template\_random\_access (_IIter __begin, _IIter __end,`  
`_UserOp __user_op, _Functionality &__functionality, _Red __reduction, _-`  
`Result __reduction_start, _Result &__output, typename std::iterator_traits< _-`  
`IIter >::difference_type __bound, \_Parallelism __parallelism_tag)`
- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result`  
`>`  
`_Op \_\_for\_each\_template\_random\_access\_ed (_RAIter __begin, _RAIter __-`  
`end, _Op __o, _Fu &__f, _Red __r, _Result __base, _Result &__output, type-`  
`name std::iterator_traits< _RAIter >::difference_type __bound)`
- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result`  
`>`  
`_Op \_\_for\_each\_template\_random\_access\_omp\_loop (_RAIter __begin, _-`  
`RAIter __end, _Op __o, _Fu &__f, _Red __r, _Result __base, _Result &__-`  
`output, typename std::iterator_traits< _RAIter >::difference_type __bound)`
- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result`  
`>`  
`_Op \_\_for\_each\_template\_random\_access\_omp\_loop\_static (_RAIter __begin,`  
`_RAIter __end, _Op __o, _Fu &__f, _Red __r, _Result __base, _Result &__-`  
`output, typename std::iterator_traits< _RAIter >::difference_type __bound)`



- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>`  
`>`  
`_Op __for_each_template_random_access_workstealing (_RAIter __begin, _`  
`RAIter __end, _Op __op, _Fu &__f, _Red __r, _Result __base, _Result &__`  
`output, typename std::iterator_traits< _RAIter >::difference_type __bound)`
- `__ThreadIndex __get_max_threads ()`
- `bool __is_parallel (const __Parallelism __p)`
- `template<typename _Iter, typename _Compare >`  
`bool __is_sorted (_Iter __begin, _Iter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare >`  
`_RAIter __median_of_three_iterators (_RAIter __a, _RAIter __b, _RAIter __c,`  
`_Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _`  
`DifferenceTp, typename _Compare >`  
`_OutputIterator __merge_advance (_RAIter1 &__begin1, _RAIter1 __end1, _`  
`RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp`  
`__max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _`  
`DifferenceTp, typename _Compare >`  
`_OutputIterator __merge_advance_movc (_RAIter1 &__begin1, _RAIter1 _`  
`__end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __target, _`  
`DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _`  
`DifferenceTp, typename _Compare >`  
`_OutputIterator __merge_advance_usual (_RAIter1 &__begin1, _RAIter1 _`  
`__end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __target, _`  
`DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _Compare >`  
`_RAIter3 __parallel_merge_advance (_RAIter1 &__begin1, _RAIter1 __`  
`end1, _RAIter2 &__begin2, _RAIter2 __end2, _RAIter3 __target, typename`  
`std::iterator_traits< _RAIter1 >::difference_type __max_length, _Compare _`  
`__comp)`
- `template<typename _RAIter1, typename _RAIter3, typename _Compare >`  
`_RAIter3 __parallel_merge_advance (_RAIter1 &__begin1, _RAIter1 __`  
`end1, _RAIter1 &__begin2, _RAIter1 __end2, _RAIter3 __target, typename`  
`std::iterator_traits< _RAIter1 >::difference_type __max_length, _Compare _`  
`__comp)`
- `template<typename _RAIter, typename _Compare >`  
`void __parallel_nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end,`  
`_Compare __comp)`
- `template<typename _RAIter, typename _Compare >`  
`void __parallel_partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __`  
`end, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator __parallel_partial_sum (_Iter __begin, _Iter __end, _`  
`OutputIterator __result, _BinaryOperation __bin_op)`

- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator \_\_parallel\_partial\_sum\_basecase (_Iter __begin, _Iter __end,`  
`_OutputIterator __result, _BinaryOperation __bin_op, typename std::iterator_`  
`traits< _Iter >::value_type __value)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator \_\_parallel\_partial\_sum\_linear (_Iter __begin, _Iter __end, _`  
`OutputIterator __result, _BinaryOperation __bin_op, typename std::iterator_`  
`traits< _Iter >::difference_type __n)`
- `template<typename _RAIter, typename _Predicate >`  
`std::iterator_traits< _RAIter >::difference_type \_\_parallel\_partition (_RAIter`  
`__begin, _RAIter __end, _Predicate __pred, \_ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void \_\_parallel\_random\_shuffle (_RAIter __begin, _RAIter __end, _`  
`RandomNumberGenerator __rng=\_RandomNumber())`
- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void \_\_parallel\_random\_shuffle\_drs (_RAIter __begin, _RAIter __end, type-`  
`name std::iterator_traits< _RAIter >::difference_type __n, \_ThreadIndex __`  
`num_threads, _RandomNumberGenerator &__rng)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void \_\_parallel\_random\_shuffle\_drs\_pu (\_DRSSorterPU< _RAIter, _`  
`RandomNumberGenerator > *__pus)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`  
`_OutputIterator \_\_parallel\_set\_difference (_Iter __begin1, _Iter __end1, _`  
`Iter __begin2, _Iter __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`  
`_OutputIterator \_\_parallel\_set\_intersection (_Iter __begin1, _Iter __end1, _`  
`Iter __begin2, _Iter __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Operation >`  
`_OutputIterator \_\_parallel\_set\_operation (_Iter __begin1, _Iter __end1, _`  
`Iter __begin2, _Iter __end2, _OutputIterator __result, _Operation __op)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`  
`_OutputIterator \_\_parallel\_set\_symmetric\_difference (_Iter __begin1, _Iter`  
`__end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _Compare __`  
`comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`  
`_OutputIterator \_\_parallel\_set\_union (_Iter __begin1, _Iter __end1, _Iter _`  
`__begin2, _Iter __end2, _OutputIterator __result, _Compare __comp)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void \_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp,`  
`parallel\_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void \_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp,`  
`balanced\_quicksort\_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void \_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp,`  
`multiway\_mergesort\_sampling\_tag __parallelism)`

- `template<bool __stable, typename _RAIter, typename _Compare, typename _Parallelism >`  
`void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_exact_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, quicksort_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, default_parallel_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`  
`void __parallel_sort_qs (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >`  
`void __parallel_sort_qs_conquer (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >`  
`std::iterator_traits< _RAIter >::difference_type __parallel_sort_qs_divide (_RAIter __begin, _RAIter __end, _Compare __comp, typename std::iterator_traits< _RAIter >::difference_type __pivot_rank, typename std::iterator_traits< _RAIter >::difference_type __num_samples, _ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >`  
`void __parallel_sort_qsb (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)`
- `template<typename _Iter, class _OutputIterator >`  
`_OutputIterator __parallel_unique_copy (_Iter __first, _Iter __last, _OutputIterator __result)`
- `template<typename _Iter, class _OutputIterator, class _BinaryPredicate >`  
`_OutputIterator __parallel_unique_copy (_Iter __first, _Iter __last, _OutputIterator __result, _BinaryPredicate __binary_pred)`
- `template<typename _RAIter, typename _Compare >`  
`void __qsb_conquer (_QSBThreadLocal< _RAIter > *__tls, _RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __iam, _ThreadIndex __num_threads, bool __parent_wait)`
- `template<typename _RAIter, typename _Compare >`  
`std::iterator_traits< _RAIter >::difference_type __qsb_divide (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >`  
`void __qsb_local_sort_with_helping (_QSBThreadLocal< _RAIter > *__tls, _Compare &__comp, _ThreadIndex __iam, bool __wait)`

- `template<typename _RandomNumberGenerator >`  
`int __random_number_pow2 (int __logp, _RandomNumberGenerator &__rng)`
- `template<typename _Size >`  
`_Size __rd_log2 (_Size __n)`
- `template<typename _Tp >`  
`_Tp __round_up_to_pow2 (_Tp __x)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`  
`__RAIter1 __search_template (__RAIter1 __begin1, __RAIter1 __end1, __RAIter2 __begin2, __RAIter2 __end2, _Pred __pred)`
- `template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >`  
`_RAIter3 __sequential_multiway_merge (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void __sequential_random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &__rng)`
- `template<typename _Iter >`  
`void __shrink (std::vector< _Iter > &__os_starts, size_t &__count_to_two, size_t &__range_length)`
- `template<typename _Iter >`  
`void __shrink_and_double (std::vector< _Iter > &__os_starts, size_t &__count_to_two, size_t &__range_length, const bool __make_twice)`
- `void __yield ()`
- `template<typename _DifferenceType, typename _OutputIterator >`  
`_OutputIterator equally_split (_DifferenceType __n, _ThreadIndex __num_threads, _OutputIterator __s)`
- `template<typename _DifferenceType >`  
`_DifferenceType equally_split_point (_DifferenceType __n, _ThreadIndex __num_threads, _ThreadIndex __thread_no)`
- `template<typename _Iter, typename _FunctorType >`  
`size_t list_partition (const _Iter __begin, const _Iter __end, _Iter *__starts, size_t *__lengths, const int __num_parts, _FunctorType &__f, int __oversampling=0)`
- `template<typename _Tp >`  
`const _Tp &max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp >`  
`const _Tp &min (const _Tp &__a, const _Tp &__b)`
- `template<typename _RanSeqs, typename _RankType, typename _RankIterator, typename _Compare >`  
`void multiseq_partition (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankIterator __begin_offsets, _Compare __comp=std::less< typename std::iterator_traits< typename std::iterator_traits< _RanSeqs >::value_type::first_type >::value_type >())`

- `template<typename _Tp, typename _RanSeqs, typename _RankType, typename _Compare >`  
`_Tp multiway_selection` (`_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _`  
`RankType __rank, _RankType &__offset, _Compare __comp=std::less< _Tp`  
`>()`)
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp,`  
`typename _Compare >`  
`_RAIterOut multiway_merge` (`_RAIterPairIterator __seqs_begin, _`  
`RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length,`  
`_Compare __comp, __gnu_parallel::sequential_tag`)
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp,`  
`typename _Compare >`  
`_RAIterOut multiway_merge` (`_RAIterPairIterator __seqs_begin, _`  
`RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length,`  
`_Compare __comp, default_parallel_tag __tag`)
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp,`  
`typename _Compare >`  
`_RAIterOut multiway_merge` (`_RAIterPairIterator __seqs_begin, _`  
`RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length,`  
`_Compare __comp, parallel_tag __tag=parallel_tag(0)`)
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp,`  
`typename _Compare >`  
`_RAIterOut multiway_merge` (`_RAIterPairIterator __seqs_begin, _`  
`RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length,`  
`_Compare __comp, __gnu_parallel::exact_tag __tag`)
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp,`  
`typename _Compare >`  
`_RAIterOut multiway_merge` (`_RAIterPairIterator __seqs_begin, _`  
`RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length,`  
`_Compare __comp, __gnu_parallel::sampling_tag __tag`)
- `template<template< typename RAI, typename C > class iterator, typename _RAIterIterator,`  
`typename _RAIter3, typename _DifferenceTp, typename _Compare >`  
`_RAIter3 multiway_merge_3_variant` (`_RAIterIterator __seqs_begin, _`  
`RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length,`  
`_Compare __comp`)
- `template<template< typename RAI, typename C > class iterator, typename _RAIterIterator,`  
`typename _RAIter3, typename _DifferenceTp, typename _Compare >`  
`_RAIter3 multiway_merge_4_variant` (`_RAIterIterator __seqs_begin, _`  
`RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length,`  
`_Compare __comp`)
- `template<bool __stable, typename _RAIterIterator, typename _Compare, typename _`  
`DifferenceType >`  
`void multiway_merge_exact_splitting` (`_RAIterIterator __seqs_begin, _`  
`RAIterIterator __seqs_end, _DifferenceType __length, _DifferenceType`  
`__total_length, _Compare __comp, std::vector< std::pair< _DifferenceType,`  
`_DifferenceType > > *__pieces`)

- `template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >`  
`_RAIter3 multiway_merge_loser_tree (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`
- `template<typename UnguardedLoserTree, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >`  
`_RAIter3 multiway_merge_loser_tree_sentinel (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >`  
`_RAIter3 multiway_merge_loser_tree_unguarded (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType >`  
`void multiway_merge_sampling_splitting (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, _DifferenceType > > *__pieces)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`

- `_RAIterOut` **multiway\_merge\_sentinels** (`_RAIterPairIterator` \_\_seqs\_begin, `_RAIterPairIterator` \_\_seqs\_end, `_RAIterOut` \_\_target, `_DifferenceTp` \_\_length, `_Compare` \_\_comp, [sampling\\_tag](#) \_\_tag)
- `template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Splitter, typename _Compare >`  
`_RAIter3` [parallel\\_multiway\\_merge](#) (`_RAIterIterator` \_\_seqs\_begin, `_RAIterIterator` \_\_seqs\_end, `_RAIter3` \_\_target, `_Splitter` \_\_splitter, `_DifferenceTp` \_\_length, `_Compare` \_\_comp, [ThreadIndex](#) \_\_num\_threads)
  - `template<bool __stable, bool __exact, typename _RAIter, typename _Compare >`  
`void` [parallel\\_sort\\_mwms](#) (`_RAIter` \_\_begin, `_RAIter` \_\_end, `_Compare` \_\_comp, [ThreadIndex](#) \_\_num\_threads)
  - `template<bool __stable, bool __exact, typename _RAIter, typename _Compare >`  
`void` [parallel\\_sort\\_mwms\\_pu](#) (`_PMWMSortingData<_RAIter>` \*\_\_sd, `_Compare` &\_\_comp)
  - `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut` **stable\_multiway\_merge** (`_RAIterPairIterator` \_\_seqs\_begin, `_RAIterPairIterator` \_\_seqs\_end, `_RAIterOut` \_\_target, `_DifferenceTp` \_\_length, `_Compare` \_\_comp, [parallel\\_tag](#) \_\_tag=[parallel\\_tag](#)(0))
  - `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut` **stable\_multiway\_merge** (`_RAIterPairIterator` \_\_seqs\_begin, `_RAIterPairIterator` \_\_seqs\_end, `_RAIterOut` \_\_target, `_DifferenceTp` \_\_length, `_Compare` \_\_comp, [sampling\\_tag](#) \_\_tag)
  - `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut` **stable\_multiway\_merge** (`_RAIterPairIterator` \_\_seqs\_begin, `_RAIterPairIterator` \_\_seqs\_end, `_RAIterOut` \_\_target, `_DifferenceTp` \_\_length, `_Compare` \_\_comp, [default\\_parallel\\_tag](#) \_\_tag)
  - `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut` **stable\_multiway\_merge** (`_RAIterPairIterator` \_\_seqs\_begin, `_RAIterPairIterator` \_\_seqs\_end, `_RAIterOut` \_\_target, `_DifferenceTp` \_\_length, `_Compare` \_\_comp, [\\_\\_gnu\\_parallel::sequential\\_tag](#))
  - `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut` **stable\_multiway\_merge** (`_RAIterPairIterator` \_\_seqs\_begin, `_RAIterPairIterator` \_\_seqs\_end, `_RAIterOut` \_\_target, `_DifferenceTp` \_\_length, `_Compare` \_\_comp, [\\_\\_gnu\\_parallel::exact\\_tag](#) \_\_tag)
  - `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut` **stable\_multiway\_merge\_sentinels** (`_RAIterPairIterator` \_\_seqs\_begin, `_RAIterPairIterator` \_\_seqs\_end, `_RAIterOut` \_\_target, `_DifferenceTp` \_\_length, `_Compare` \_\_comp, [\\_\\_gnu\\_parallel::exact\\_tag](#) \_\_tag)

- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_`  
`begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __`  
`_length, _Compare __comp, default\_parallel\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_`  
`begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __`  
`_length, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_`  
`begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __`  
`_length, _Compare __comp, sampling\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_`  
`begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __`  
`_length, _Compare __comp, parallel\_tag __tag=parallel\_tag(0))`

## Variables

- `static const int \_CASable\_bits`
- `static const \_CASable \_CASable\_mask`

### 4.6.1 Detailed Description

GNU parallel code for public use.

### 4.6.2 Typedef Documentation

#### 4.6.2.1 `typedef unsigned short __gnu_parallel::_BinIndex`

Type to hold the index of a bin.

Since many variables of this type are allocated, it should be chosen as small as possible.

Definition at line 47 of file `random_shuffle.h`.

#### 4.6.2.2 `typedef int64_t __gnu_parallel::_CASable`



Longest compare-and-swappable integer type on this platform.

Definition at line 127 of file `types.h`.

#### 4.6.2.3 `typedef uint64_t __gnu_parallel::_SequenceIndex`

Unsigned integer to index `__elements`. The total number of elements for each algorithm must fit into this type.

Definition at line 117 of file `types.h`.

#### 4.6.2.4 `typedef uint16_t __gnu_parallel::_ThreadIndex`

Unsigned integer to index a thread number. The maximum thread number (for each processor) must fit into this type.

Definition at line 123 of file `types.h`.

### 4.6.3 Enumeration Type Documentation

#### 4.6.3.1 `enum __gnu_parallel::_AlgorithmStrategy`

Strategies for run-time algorithm selection:

Definition at line 67 of file `types.h`.

#### 4.6.3.2 `enum __gnu_parallel::_FindAlgorithm`

Find algorithms:

Definition at line 106 of file `types.h`.

#### 4.6.3.3 `enum __gnu_parallel::_MultiwayMergeAlgorithm`

Merging algorithms:

Definition at line 85 of file `types.h`.

#### 4.6.3.4 `enum __gnu_parallel::_Parallelism`

Run-time equivalents for the compile-time tags.

**Enumerator:**

*sequential* Not parallel.

*parallel\_unbalanced* Parallel unbalanced (equal-sized chunks).

*parallel\_balanced* Parallel balanced (work-stealing).

*parallel\_omp\_loop* Parallel with OpenMP dynamic load-balancing.

*parallel\_omp\_loop\_static* Parallel with OpenMP static load-balancing.

*parallel\_taskqueue* Parallel with OpenMP taskqueue construct.

Definition at line 44 of file `types.h`.

#### 4.6.3.5 `enum __gnu_parallel::_PartialSumAlgorithm`

Partial sum algorithms: recursive, linear.

Definition at line 91 of file `types.h`.

#### 4.6.3.6 `enum __gnu_parallel::_SortAlgorithm`

Sorting algorithms:

Definition at line 76 of file `types.h`.

#### 4.6.3.7 `enum __gnu_parallel::_SplittingAlgorithm`

Sorting/merging algorithms: sampling, `__exact`.

Definition at line 98 of file `types.h`.

#### 4.6.4 Function Documentation

**4.6.4.1** `template<typename _RAIter, typename _DifferenceTp > void  
__gnu_parallel::__calc_borders ( _RAIter __elements, _DifferenceTp  
__length, _DifferenceTp * __off )`

Precalculate `__advances` for Knuth-Morris-Pratt algorithm.

##### Parameters

`__elements` Begin iterator of sequence to search for.

`__length` Length of sequence to search for.

`__advances` Returned `__offsets`.

Definition at line 51 of file `search.h`.

Referenced by `__search_template()`.

**4.6.4.2** `template<typename _Tp > bool __gnu_parallel::__compare_and_swap  
( volatile _Tp * __ptr, _Tp __comparand, _Tp __replacement )  
[inline]`

Compare `*__ptr` and `__comparand`. If equal, let `*__ptr=__replacement` and return `true`, return `false` otherwise.

Implementation is heavily platform-dependent.

##### Parameters

`__ptr` Pointer to signed integer.

`__comparand` Compare value.

`__replacement` Replacement value.

Definition at line 337 of file `parallel/compatibility.h`.

References `__compare_and_swap_32()`, and `__compare_and_swap_64()`.

Referenced by `__parallel_partition()`, `__gnu_parallel::-RestrictedBoundedConcurrentQueue< _Piece >::pop_back()`, and `__gnu_parallel::-RestrictedBoundedConcurrentQueue< _Piece >::pop_front()`.

**4.6.4.3** `bool __gnu_parallel::__compare_and_swap_32 ( volatile int32_t *  
__ptr, int32_t __comparand, int32_t __replacement ) [inline]`

Compare `*__ptr` and `__comparand`. If equal, let `*__ptr=__replacement` and return `true`, return `false` otherwise.

Implementation is heavily platform-dependent.

#### Parameters

`__ptr` Pointer to 32-bit signed integer.

`__comparand` Compare value.

`__replacement` Replacement value.

Definition at line 240 of file `parallel/compatibility.h`.

Referenced by `__compare_and_swap()`.

**4.6.4.4** `bool __gnu_parallel::__compare_and_swap_64 ( volatile int64_t *  
__ptr, int64_t __comparand, int64_t __replacement ) [inline]`

Compare `*__ptr` and `__comparand`. If equal, let `*__ptr=__replacement` and return `true`, return `false` otherwise.

Implementation is heavily platform-dependent.

#### Parameters

`__ptr` Pointer to 64-bit signed integer.

`__comparand` Compare value.

`__replacement` Replacement value.

Definition at line 283 of file `parallel/compatibility.h`.

Referenced by `__compare_and_swap()`.

**4.6.4.5** `void __gnu_parallel::__decode2 ( _CASable __x, int & __a, int &  
__b ) [inline]`

Decode two integers from one `gnu_parallel::_CASable`.

#### Parameters

`__x` [\\_\\_gnu\\_parallel::\\_CASable](#) to decode integers from.

`__a` First integer, to be decoded from the most-significant `_CASable_bits/2` bits of `__x`.

`__b` Second integer, to be encoded in the least-significant `_CASable_bits/2` bits of `__x`.

See also

[\\_\\_encode2](#)

Definition at line 133 of file `parallel/base.h`.

References `_CASable_bits`, and `_CASable_mask`.

Referenced by `__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Piece >::pop_back()`, `__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Piece >::pop_front()`, and `__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Piece >::push_front()`.

**4.6.4.6** `template<typename _RAIter, typename _DifferenceTp> void  
__gnu_parallel::_determine_samples ( _PMWMSSortingData<  
_RAIter> * __sd, _DifferenceTp __num_samples )`

Select `_M_samples` from a sequence.

Parameters

`__sd` Pointer to algorithm data. Result will be placed in `__sd->_M_samples`.  
`__num_samples` Number of `_M_samples` to select.

Definition at line 97 of file `multiway_mergesort.h`.

References `__gnu_parallel::_PMWMSSortingData< _RAIter>::_M_samples`, `__gnu_parallel::_PMWMSSortingData< _RAIter>::_M_source`, `__gnu_parallel::_PMWMSSortingData< _RAIter>::_M_starts`, and `equally_split()`.

**4.6.4.7** `_CASable __gnu_parallel::_encode2 ( int __a, int __b ) [inline]`

Encode two integers into one `gnu_parallel::_CASable`.

Parameters

`__a` First integer, to be encoded in the most-significant `_CASable_bits/2` bits.  
`__b` Second integer, to be encoded in the least-significant `_CASable_bits/2` bits.

**Returns**

value encoding `__a` and `__b`.

**See also**

[\\_\\_decode2](#)

Definition at line 119 of file `parallel/base.h`.

References `_CASable_bits`.

Referenced by `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Piece>::_RestrictedBoundedConcurrentQueue()`, `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Piece>::pop_back()`, `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Piece>::pop_front()`, and `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Piece>::push_front()`.

#### 4.6.4.8 `template<typename _Tp> _Tp __gnu_parallel::__fetch_and_add ( volatile _Tp * __ptr, _Tp __addend ) [inline]`

Add a value to a variable, atomically.

Implementation is heavily platform-dependent.

**Parameters**

`__ptr` Pointer to a signed integer.

`__addend` Value to add.

Definition at line 186 of file `parallel/compatibility.h`.

References `__fetch_and_add_32()`, and `__fetch_and_add_64()`.

Referenced by `__parallel_partition()`, and `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Piece>::push_front()`.

#### 4.6.4.9 `int32_t __gnu_parallel::__fetch_and_add_32 ( volatile int32_t * __ptr, int32_t __addend ) [inline]`

Add a value to a variable, atomically.

Implementation is heavily platform-dependent.

**Parameters**

`__ptr` Pointer to a 32-bit signed integer.

`__addend` Value to add.

Definition at line 95 of file `parallel/compatibility.h`.

Referenced by `__fetch_and_add()`.

**4.6.4.10** `int64_t __gnu_parallel::__fetch_and_add_64 ( volatile int64_t * __ptr,  
int64_t __addend ) [inline]`

Add a value to a variable, atomically.

Implementation is heavily platform-dependent.

#### Parameters

`__ptr` Pointer to a 64-bit signed integer.

`__addend` Value to add.

Definition at line 134 of file `parallel/compatibility.h`.

Referenced by `__fetch_and_add()`.

**4.6.4.11** `template<typename _RAIter1 , typename _RAIter2 , typename  
_Pred , typename _Selector > std::pair<_RAIter1, _RAIter2>  
__gnu_parallel::__find_template ( _RAIter1 __begin1, _RAIter1  
__end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector )  
[inline]`

Parallel `std::find`, switch for different algorithms.

#### Parameters

`__begin1` Begin iterator of first sequence.

`__end1` End iterator of first sequence.

`__begin2` Begin iterator of second sequence. Must have same length as first sequence.

`__pred` Find predicate.

`__selector` \_Functionality (e. g. `std::find_if()`, `std::equal()`,...)

#### Returns

Place of finding in both sequences.

Definition at line 60 of file `find.h`.

References `__gnu_parallel::_Settings::get()`, and `std::make_pair()`.

```
4.6.4.12 template<typename _RAIter1 , typename _RAIter2 , typename
 _Pred , typename _Selector > std::pair<_RAIter1, _RAIter2>
 __gnu_parallel::__find_template (_RAIter1 __begin1, _RAIter1
 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector,
 equal_split_tag)
```

Parallel `std::find`, equal splitting variant.

#### Parameters

- `__begin1` Begin iterator of first sequence.
- `__end1` End iterator of first sequence.
- `__begin2` Begin iterator of second sequence. Second \_\_sequence must have same length as first sequence.
- `__pred` Find predicate.
- `__selector` \_Functionality (e. g. `std::find_if()`, `std::equal()`,...)

#### Returns

Place of finding in both sequences.

Definition at line 97 of file `find.h`.

References `_GLIBCXX_CALL`, and `equally_split()`.

```
4.6.4.13 template<typename _RAIter1 , typename _RAIter2 , typename
 _Pred , typename _Selector > std::pair<_RAIter1, _RAIter2>
 __gnu_parallel::__find_template (_RAIter1 __begin1, _RAIter1
 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector,
 growing_blocks_tag)
```

Parallel `std::find`, growing block size variant.

#### Parameters

- `__begin1` Begin iterator of first sequence.
- `__end1` End iterator of first sequence.



`__begin2` Begin iterator of second sequence. Second `__sequence` must have same length as first sequence.

`__pred` Find predicate.

`__selector` \_Functionality (e. g. `std::find_if()`, `std::equal()`,...)

### Returns

Place of finding in both sequences.

### See also

[\\_\\_gnu\\_parallel::\\_\\_Settings::find\\_sequential\\_search\\_size](#)

[\\_\\_gnu\\_parallel::\\_\\_Settings::find\\_scale\\_factor](#)

There are two main differences between the growing blocks and the constant-size blocks variants. 1. For GB, the block size grows; for CSB, the block size is fixed. 2. For GB, the blocks are allocated dynamically; for CSB, the blocks are allocated in a predetermined manner, namely spacial round-robin.

Definition at line 185 of file `find.h`.

References `_GLIBCXX_CALL`, `__gnu_parallel::__Settings::find_scale_factor`, `__gnu_parallel::__Settings::find_sequential_search_size`, and `__gnu_parallel::__Settings::get()`.

**4.6.4.14** `template<typename _RAIter1 , typename _RAIter2 , typename _Pred , typename _Selector > std::pair<_RAIter1, _RAIter2> __gnu_parallel::__find_template ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, constant_size_blocks_tag )`

Parallel `std::find`, constant block size variant.

### Parameters

`__begin1` Begin iterator of first sequence.

`__end1` End iterator of first sequence.

`__begin2` Begin iterator of second sequence. Second `__sequence` must have same length as first sequence.

`__pred` Find predicate.

`__selector` \_Functionality (e. g. `std::find_if()`, `std::equal()`,...)

### Returns

Place of finding in both sequences.

## See also

[\\_\\_gnu\\_parallel::\\_Settings::find\\_sequential\\_search\\_size](#)

[\\_\\_gnu\\_parallel::\\_Settings::find\\_block\\_size](#) There are two main differences between the growing blocks and the constant-size blocks variants. 1. For GB, the block size grows; for CSB, the block size is fixed. 2. For GB, the blocks are allocated dynamically; for CSB, the blocks are allocated in a predetermined manner, namely spacial round-robin.

Definition at line 315 of file find.h.

References [\\_GLIBCXX\\_CALL](#), [\\_\\_gnu\\_parallel::\\_Settings::find\\_initial\\_block\\_size](#), [\\_\\_gnu\\_parallel::\\_Settings::find\\_sequential\\_search\\_size](#), and [\\_\\_gnu\\_parallel::\\_Settings::get\(\)](#).

**4.6.4.15** `template<typename _Iter , typename _UserOp , typename  
_Functionality , typename _Red , typename _Result > _UserOp  
__gnu_parallel::_for_each_template_random_access ( _Iter  
__begin, _Iter __end, _UserOp __user_op, _Functionality &  
__functionality, _Red __reduction, _Result __reduction_start, _Result  
& __output, typename std::iterator_traits< _Iter >::difference_type  
__bound, _Parallelism __parallelism_tag )`

Chose the desired algorithm by evaluating `__parallelism_tag`.

## Parameters

**`__begin`** Begin iterator of input sequence.  
**`__end`** End iterator of input sequence.  
**`__user_op`** A user-specified functor (comparator, predicate, associative operator,...)  
**`__functionality`** functor to *process* an element with `__user_op` (depends on desired functionality, e. g. accumulate, for\_each,...)  
**`__reduction`** Reduction functor.  
**`__reduction_start`** Initial value for reduction.  
**`__output`** Output iterator.  
**`__bound`** Maximum number of elements processed.  
**`__parallelism_tag`** Parallelization method

Definition at line 61 of file for\_each.h.

References [\\_\\_for\\_each\\_template\\_random\\_access\\_ed\(\)](#), [\\_\\_for\\_each\\_template\\_random\\_access\\_omp\\_loop\(\)](#), [\\_\\_for\\_each\\_template\\_random\\_access\\_workstealing\(\)](#), [parallel\\_omp\\_loop](#), [parallel\\_omp\\_loop\\_static](#), and [parallel\\_unbalanced](#).

**4.6.4.16** `template<typename _RAIter , typename _Op , typename  
_Fu , typename _Red , typename _Result > _Op  
__gnu_parallel::__for_each_template_random_access_ed ( _RAIter  
__begin, _RAIter __end, _Op __o, _Fu & __f, _Red __r, _Result  
__base, _Result & __output, typename std::iterator_traits< _RAIter  
>::difference_type __bound )`

Embarrassingly parallel algorithm for random access iterators, using hand-crafted parallelization by equal splitting the work.

#### Parameters

- `__begin` Begin iterator of element sequence.
- `__end` End iterator of element sequence.
- `__o` User-supplied functor (comparator, predicate, adding functor, ...)
- `__f` Functor to "process" an element with `__op` (depends on desired functionality, e. g. for `std::for_each()`, ...).
- `__r` Functor to "add" a single `__result` to the already processed elements (depends on functionality).
- `__base` Base value for reduction.
- `__output` Pointer to position where final result is written to
- `__bound` Maximum number of elements processed (e. g. for `std::count_n()`).

#### Returns

User-supplied functor (that may contain a part of the result).

Definition at line 67 of file `par_loop.h`.

References `equally_split_point()`.

Referenced by `__for_each_template_random_access()`.

**4.6.4.17** `template<typename _RAIter , typename _Op , typename  
_Fu , typename _Red , typename _Result > _Op  
__gnu_parallel::__for_each_template_random_access_omp_loop (   
_RAIter __begin, _RAIter __end, _Op __o, _Fu & __f, _Red __r,  
_Result __base, _Result & __output, typename std::iterator_traits<  
_RAIter >::difference_type __bound )`

Embarrassingly parallel algorithm for random access iterators, using an OpenMP for loop.

**Parameters**

- `__begin` Begin iterator of element sequence.
- `__end` End iterator of element sequence.
- `__o` User-supplied functor (comparator, predicate, adding functor, etc.).
- `__f` Functor to *process* an element with `__op` (depends on desired functionality, e. g. for `std::for_each()`, ...).
- `__r` Functor to *add* a single `__result` to the already processed elements (depends on functionality).
- `__base` Base value for reduction.
- `__output` Pointer to position where final result is written to
- `__bound` Maximum number of elements processed (e. g. for `std::count_n()`).

**Returns**

User-supplied functor (that may contain a part of the result).

Definition at line 67 of file `omp_loop.h`.

Referenced by `__for_each_template_random_access()`.

**4.6.4.18** `template<typename _RAIter , typename _Op , typename _Fu ,  
 typename _Red , typename _Result > _Op __gnu_parallel::__-  
 for_each_template_random_access_omp_loop_static ( _RAIter  
 __begin, _RAIter __end, _Op __o, _Fu & __f, _Red __r, _Result  
 __base, _Result & __output, typename std::iterator_traits< _RAIter  
 >::difference_type __bound )`

Embarrassingly parallel algorithm for random access iterators, using an OpenMP for loop with static scheduling.

**Parameters**

- `__begin` Begin iterator of element sequence.
- `__end` End iterator of element sequence.
- `__o` User-supplied functor (comparator, predicate, adding functor, ...).
- `__f` Functor to *process* an element with `__op` (depends on desired functionality, e. g. for `std::for_each()`, ...).
- `__r` Functor to *add* a single `__result` to the already processed `__elements` (depends on functionality).
- `__base` Base value for reduction.

`__output` Pointer to position where final result is written to  
`__bound` Maximum number of elements processed (e. g. for `std::count_n()`).

**Returns**

User-supplied functor (that may contain a part of the result).

Definition at line 66 of file `omp_loop_static.h`.

**4.6.4.19** `template<typename _RAIter, typename _Op, typename  
_Fu, typename _Red, typename _Result > _Op  
__gnu_parallel::__for_each_template_random_access_workstealing (  
_RAIter __begin, _RAIter __end, _Op __op, _Fu & __f, _Red __r,  
_Result __base, _Result & __output, typename std::iterator_traits<  
_RAIter >::difference_type __bound )`

Work stealing algorithm for random access iterators.

Uses O(1) additional memory. Synchronization at job lists is done with atomic operations.

**Parameters**

`__begin` Begin iterator of element sequence.  
`__end` End iterator of element sequence.  
`__op` User-supplied functor (comparator, predicate, adding functor, ...).  
`__f` Functor to *process* an element with `__op` (depends on desired functionality, e. g. for `std::for_each()`, ...).  
`__r` Functor to *add* a single `__result` to the already processed elements (depends on functionality).  
`__base` Base value for reduction.  
`__output` Pointer to position where final result is written to  
`__bound` Maximum number of elements processed (e. g. for `std::count_n()`).

**Returns**

User-supplied functor (that may contain a part of the result).

Definition at line 99 of file `workstealing.h`.

References `__yield()`, `_GLIBCXX_CALL`, `__gnu_parallel::Job< _DifferenceTp >::_M_first`, `__gnu_parallel::Job< _DifferenceTp >::_M_last`, `__gnu_parallel::Job< _DifferenceTp >::_M_load`, `__gnu_parallel::Settings::cache_line_size`, `__gnu_parallel::Settings::get()`, and `min()`.

Referenced by `__for_each_template_random_access()`.

**4.6.4.20** `template<typename _Iter , typename _Compare > bool  
__gnu_parallel::__is_sorted ( _Iter __begin, _Iter __end,  
_Compare __comp )`

Check whether [`__begin`, `__end`) is sorted according to `__comp`.

#### Parameters

`__begin` Begin iterator of sequence.  
`__end` End iterator of sequence.  
`__comp` Comparator.

#### Returns

`true` if sorted, `false` otherwise.

Definition at line 51 of file `checkers.h`.

Referenced by `__sequential_multiway_merge()`, `multiway_merge_loser_tree_sentinel()`, and `parallel_multiway_merge()`.

**4.6.4.21** `template<typename _RAIter , typename _Compare > _RAIter  
__gnu_parallel::__median_of_three_iterators ( _RAIter __a, _RAIter  
__b, _RAIter __c, _Compare __comp )`

Compute the median of three referenced elements, according to `__comp`.

#### Parameters

`__a` First iterator.  
`__b` Second iterator.  
`__c` Third iterator.  
`__comp` Comparator.

Definition at line 398 of file `parallel/base.h`.

Referenced by `__qsb_divide()`.

**4.6.4.22** `template<typename _RAIter1 , typename _RAIter2 , typename  
_OutputIterator , typename _DifferenceTp , typename _Compare  
> _OutputIterator __gnu_parallel::__merge_advance ( _RAIter1  
& __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2  
__end2, _OutputIterator __target, _DifferenceTp __max_length,  
_Compare __comp ) [inline]`

Merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant. Static switch on whether to use the conditional-move variant.

#### Parameters

`__begin1` Begin iterator of first sequence.  
`__end1` End iterator of first sequence.  
`__begin2` Begin iterator of second sequence.  
`__end2` End iterator of second sequence.  
`__target` Target begin iterator.  
`__max_length` Maximum number of elements to merge.  
`__comp` Comparator.

#### Returns

Output end iterator.

Definition at line 171 of file `merge.h`.

References `__merge_advance_movc()`, and `_GLIBCXX_CALL`.

Referenced by `__parallel_merge_advance()`, and `__sequential_multiway_merge()`.

**4.6.4.23** `template<typename _RAIter1 , typename _RAIter2 , typename  
_OutputIterator , typename _DifferenceTp , typename _Compare >  
_OutputIterator __gnu_parallel::__merge_advance_movc ( _RAIter1  
& __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2  
__end2, _OutputIterator __target, _DifferenceTp __max_length,  
_Compare __comp )`

Merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant. Specially designed code should allow the compiler to generate conditional moves instead of branches.

**Parameters**

- `__begin1` Begin iterator of first sequence.
- `__end1` End iterator of first sequence.
- `__begin2` Begin iterator of second sequence.
- `__end2` End iterator of second sequence.
- `__target` Target begin iterator.
- `__max_length` Maximum number of elements to merge.
- `__comp` Comparator.

**Returns**

Output end iterator.

Definition at line 105 of file `merge.h`.

Referenced by `__merge_advance()`.

**4.6.4.24** `template<typename _RAIter1 , typename _RAIter2 , typename  
_OutputIterator , typename _DifferenceTp , typename _Compare >  
_OutputIterator __gnu_parallel::__merge_advance_usual ( _RAIter1  
& __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2  
__end2, _OutputIterator __target, _DifferenceTp __max_length,  
_Compare __comp )`

Merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant.

**Parameters**

- `__begin1` Begin iterator of first sequence.
- `__end1` End iterator of first sequence.
- `__begin2` Begin iterator of second sequence.
- `__end2` End iterator of second sequence.
- `__target` Target begin iterator.
- `__max_length` Maximum number of elements to merge.
- `__comp` Comparator.

**Returns**

Output end iterator.

Definition at line 57 of file `merge.h`.



```

4.6.4.25 template<typename _RAIter1 , typename _RAIter2 ,
 typename _RAIter3 , typename _Compare > _RAIter3
 __gnu_parallel::__parallel_merge_advance (_RAIter1 & __begin1,
 _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2 __end2,
 _RAIter3 __target, typename std::iterator_traits< _RAIter1
 >::difference_type __max_length, _Compare __comp) [inline]

```

Merge routine fallback to sequential in case the iterators of the two input sequences are of different type.

#### Parameters

`__begin1` Begin iterator of first sequence.  
`__end1` End iterator of first sequence.  
`__begin2` Begin iterator of second sequence.  
`__end2` End iterator of second sequence.  
`__target` Target begin iterator.  
`__max_length` Maximum number of elements to merge.  
`__comp` Comparator.

#### Returns

Output end iterator.

Definition at line 195 of file `merge.h`.

References `__merge_advance()`.

```

4.6.4.26 template<typename _RAIter1 , typename _RAIter3 , typename
 _Compare > _RAIter3 __gnu_parallel::__parallel_merge_advance (
 _RAIter1 & __begin1, _RAIter1 __end1, _RAIter1 & __begin2,
 _RAIter1 __end2, _RAIter3 __target, typename std::iterator_traits<
 _RAIter1 >::difference_type __max_length, _Compare __comp)
 [inline]

```

Parallel merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant. The functionality is projected onto `parallel_multiway_merge`.

**Parameters**

- `__begin1` Begin iterator of first sequence.
- `__end1` End iterator of first sequence.
- `__begin2` Begin iterator of second sequence.
- `__end2` End iterator of second sequence.
- `__target` Target begin iterator.
- `__max_length` Maximum number of elements to merge.
- `__comp` Comparator.

**Returns**

Output end iterator.

Definition at line 223 of file `merge.h`.

References `std::make_pair()`, `multiway_merge_exact_splitting()`, and `parallel_multiway_merge()`.

**4.6.4.27** `template<typename _RAIter, typename _Compare> void  
__gnu_parallel::__parallel_nth_element ( _RAIter __begin, _RAIter  
__nth, _RAIter __end, _Compare __comp )`

Parallel implementation of `std::nth_element()`.

**Parameters**

- `__begin` Begin iterator of input sequence.
- `__nth` Iterator of element that must be in position afterwards.
- `__end` End iterator of input sequence.
- `__comp` Comparator.

Definition at line 332 of file `partition.h`.

References `__parallel_partition()`, `_GLIBCXX_CALL`, `__gnu_parallel::_Settings::get()`, `max()`, `__gnu_parallel::_Settings::nth_element_minimal_n`, and `__gnu_parallel::_Settings::partition_minimal_n`.

Referenced by `__parallel_partial_sort()`.

**4.6.4.28** `template<typename _RAIter , typename _Compare > void  
 __gnu_parallel::__parallel_partial_sort ( _RAIter __begin, _RAIter  
 __middle, _RAIter __end, _Compare __comp )`

Parallel implementation of `std::partial_sort()`.

#### Parameters

`__begin` Begin iterator of input sequence.  
`__middle` Sort until this position.  
`__end` End iterator of input sequence.  
`__comp` Comparator.

Definition at line 422 of file `partition.h`.

References `__parallel_nth_element()`.

**4.6.4.29** `template<typename _IIter , typename _OutputIterator  
 , typename _BinaryOperation > _OutputIterator  
 __gnu_parallel::__parallel_partial_sum ( _IIter __begin, _IIter  
 __end, _OutputIterator __result, _BinaryOperation __bin_op )`

Parallel partial sum front-`__end`.

#### Parameters

`__begin` Begin iterator of input sequence.  
`__end` End iterator of input sequence.  
`__result` Begin iterator of output sequence.  
`__bin_op` Associative binary function.

#### Returns

End iterator of output sequence.

Definition at line 203 of file `partial_sum.h`.

References `__parallel_partial_sum_linear()`, `_GLIBCXX_CALL`, and `__gnu_parallel::_Settings::get()`.

**4.6.4.30** `template<typename _Iter , typename _OutputIterator  
, typename _BinaryOperation > _OutputIterator  
__gnu_parallel::__parallel_partial_sum_basecase ( _Iter __begin,  
_Iter __end, _OutputIterator __result, _BinaryOperation __bin_op,  
typename std::iterator_traits< _Iter >::value_type __value )`

Base case prefix sum routine.

#### Parameters

`__begin` Begin iterator of input sequence.  
`__end` End iterator of input sequence.  
`__result` Begin iterator of output sequence.  
`__bin_op` Associative binary function.  
`__value` Start value. Must be passed since the neutral element is unknown in general.

#### Returns

End iterator of output sequence.

Definition at line 58 of file `partial_sum.h`.

Referenced by `__parallel_partial_sum_linear()`.

**4.6.4.31** `template<typename _Iter , typename _OutputIterator  
, typename _BinaryOperation > _OutputIterator  
__gnu_parallel::__parallel_partial_sum_linear ( _Iter __begin,  
_Iter __end, _OutputIterator __result, _BinaryOperation __bin_op,  
typename std::iterator_traits< _Iter >::difference_type __n )`

Parallel partial sum implementation, two-phase approach, no recursion.

#### Parameters

`__begin` Begin iterator of input sequence.  
`__end` End iterator of input sequence.  
`__result` Begin iterator of output sequence.  
`__bin_op` Associative binary function.  
`__n` Length of sequence.  
`__num_threads` Number of threads to use.

**Returns**

End iterator of output sequence.

Definition at line 90 of file partial\_sum.h.

References \_\_parallel\_partial\_sum\_basecase(), equally\_split(), \_\_gnu\_parallel::\_Settings::get(), and \_\_gnu\_parallel::\_Settings::partial\_sum\_dilation.

Referenced by \_\_parallel\_partial\_sum().

```
4.6.4.32 template<typename _RAIter , typename _Predicate
> std::iterator_traits<_RAIter>::difference_type
 __gnu_parallel::__parallel_partition (_RAIter __begin, _RAIter
 __end, _Predicate __pred, _ThreadIndex __num_threads)
```

Parallel implementation of std::partition.

**Parameters**

*\_\_begin* Begin iterator of input sequence to split.

*\_\_end* End iterator of input sequence to split.

*\_\_pred* Partition predicate, possibly including some kind of pivot.

*\_\_num\_threads* Maximum number of threads to use for this task.

**Returns**

Number of elements not fulfilling the predicate.

Definition at line 56 of file partition.h.

References \_\_compare\_and\_swap(), \_\_fetch\_and\_add(), \_GLIBCXX\_CALL, \_GLIBCXX\_VOLATILE, \_\_gnu\_parallel::\_Settings::get(), \_\_gnu\_parallel::\_Settings::partition\_chunk\_share, and \_\_gnu\_parallel::\_Settings::partition\_chunk\_size.

Referenced by \_\_parallel\_nth\_element(), \_\_parallel\_sort\_qs\_divide(), and \_\_qsb\_divide().

```
4.6.4.33 template<typename _RAIter , typename _RandomNumberGenerator
> void __gnu_parallel::__parallel_random_shuffle (_RAIter
 __begin, _RAIter __end, _RandomNumberGenerator __rng =
 _RandomNumber()) [inline]
```

Parallel random public call.

**Parameters**

- \_\_begin** Begin iterator of sequence.
- \_\_end** End iterator of sequence.
- \_\_rng** Random number generator to use.

Definition at line 517 of file random\_shuffle.h.

References \_\_parallel\_random\_shuffle\_drs().

**4.6.4.34** `template<typename _RAIter, typename _RandomNumberGenerator  
> void __gnu_parallel::__parallel_random_shuffle_drs ( _RAIter  
__begin, _RAIter __end, typename std::iterator_traits<  
_RAIter >::difference_type __n, _ThreadIndex __num_threads,  
_RandomNumberGenerator & __rng )`

Main parallel random shuffle step.

**Parameters**

- \_\_begin** Begin iterator of sequence.
- \_\_end** End iterator of sequence.
- \_\_n** Length of sequence.
- \_\_num\_threads** Number of threads to use.
- \_\_rng** Random number generator to use.

Definition at line 263 of file random\_shuffle.h.

References \_\_gnu\_parallel::\_DRSSorterPU< \_RAIter, \_RandomNumberGenerator >::\_bins\_end, \_\_parallel\_random\_shuffle\_drs\_pu(), \_\_rd\_log2(), \_\_round\_up\_to\_pow2(), \_\_sequential\_random\_shuffle(), \_GLIBCXX\_CALL, \_\_gnu\_parallel::\_DRandomShufflingGlobalData< \_RAIter >::\_M\_bin\_proc, \_\_gnu\_parallel::\_DRSSorterPU< \_RAIter, \_RandomNumberGenerator >::\_M\_bins\_begin, \_\_gnu\_parallel::\_DRandomShufflingGlobalData< \_RAIter >::\_M\_dist, \_\_gnu\_parallel::\_DRandomShufflingGlobalData< \_RAIter >::\_M\_num\_bins, \_\_gnu\_parallel::\_DRandomShufflingGlobalData< \_RAIter >::\_M\_num\_bits, \_\_gnu\_parallel::\_DRSSorterPU< \_RAIter, \_RandomNumberGenerator >::\_M\_num\_threads, \_\_gnu\_parallel::\_DRSSorterPU< \_RAIter, \_RandomNumberGenerator >::\_M\_sd, \_\_gnu\_parallel::\_DRSSorterPU< \_RAIter, \_RandomNumberGenerator >::\_M\_seed, \_\_gnu\_parallel::\_DRandomShufflingGlobalData< \_RAIter >::\_M\_starts, \_\_gnu\_parallel::\_DRandomShufflingGlobalData< \_RAIter >::\_M\_temporaries, \_\_gnu\_parallel::\_Settings::L2\_cache\_size, std::min(), and \_\_gnu\_parallel::\_Settings::TLB\_size.

Referenced by \_\_parallel\_random\_shuffle().

**4.6.4.35** `template<typename _RAIter, typename _RandomNumberGenerator  
> void __gnu_parallel::__parallel_random_shuffle_drs_pu (`  
`_DRSSorterPU< _RAIter, _RandomNumberGenerator > * __pus )`

Random shuffle code executed by each thread.

#### Parameters

`__pus` Array of thread-local data records.

Definition at line 122 of file random\_shuffle.h.

References `__random_number_pow2()`, `__gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_dist`, `__gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_num_bins`, `__gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_num_bits`, `__gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_num_threads`, `__gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_sd`, `__gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_seed`, and `__gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_starts`.

Referenced by `__parallel_random_shuffle_drs()`.

**4.6.4.36** `template<bool __stable, typename _RAIter, typename _Compare >`  
`void __gnu_parallel::__parallel_sort ( _RAIter __begin, _RAIter`  
`__end, _Compare __comp, parallel_tag __parallelism ) [inline]`

Choose a parallel sorting algorithm.

#### Parameters

`__begin` Begin iterator of input sequence.

`__end` End iterator of input sequence.

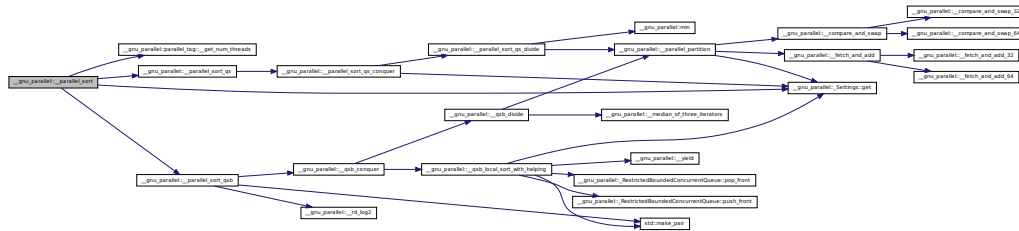
`__comp` Comparator.

`__stable` Sort `__stable`.

Definition at line 198 of file sort.h.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, `__parallel_sort_qs()`, `__parallel_sort_qsb()`, `_GLIBCXX_CALL`, and `__gnu_parallel::_Settings::get()`.

Here is the call graph for this function:



```
4.6.4.37 template<bool __stable, typename _RAIter, typename _Compare >
void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter
__end, _Compare __comp, balanced_quicksort_tag __parallelism)
[inline]
```

Choose balanced quicksort for parallel sorting.

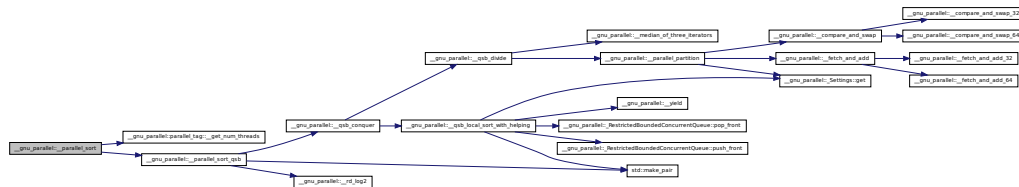
#### Parameters

- \_\_begin** Begin iterator of input sequence.
- \_\_end** End iterator of input sequence.
- \_\_comp** Comparator.
- \_\_stable** Sort \_\_stable.

Definition at line 157 of file sort.h.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, `__parallel_sort_qs()`, and `_GLIBCXX_CALL`.

Here is the call graph for this function:





**4.6.4.38** `template<bool __stable, typename _RAIter, typename _Compare >  
 void __gnu_parallel::__parallel_sort ( _RAIter __begin, _RAIter  
 __end, _Compare __comp, multiway_mergesort_sampling_tag  
 __parallelism ) [inline]`

Choose multiway mergesort with splitting by sampling, for parallel sorting.

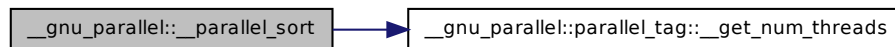
#### Parameters

***\_\_begin*** Begin iterator of input sequence.  
***\_\_end*** End iterator of input sequence.  
***\_\_comp*** Comparator.

Definition at line 117 of file sort.h.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, and `_GLIBCXX_CALL`.

Here is the call graph for this function:



**4.6.4.39** `template<bool __stable, typename _RAIter, typename _Compare >  
 void __gnu_parallel::__parallel_sort ( _RAIter __begin, _RAIter  
 __end, _Compare __comp, multiway_mergesort_tag __parallelism )  
 [inline]`

Choose multiway mergesort, splitting variant at run-time, for parallel sorting.

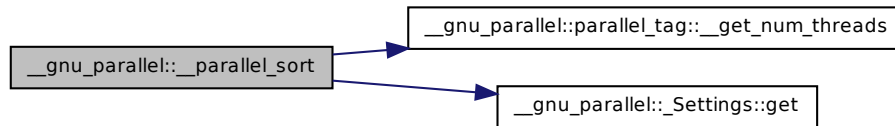
#### Parameters

***\_\_begin*** Begin iterator of input sequence.  
***\_\_end*** End iterator of input sequence.  
***\_\_comp*** Comparator.

Definition at line 74 of file sort.h.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, `_GLIBCXX_CALL`, and `__gnu_parallel::_Settings::get()`.

Here is the call graph for this function:



**4.6.4.40** `template<bool __stable, typename _RAIter, typename _Compare >`  
`void __gnu_parallel::__parallel_sort ( _RAIter __begin, _RAIter`  
`__end, _Compare __comp, multiway_mergesort_exact_tag`  
`__parallelism ) [inline]`

Choose multiway mergesort with exact splitting, for parallel sorting.

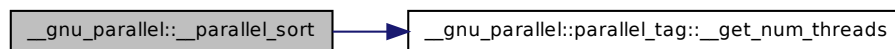
#### Parameters

- `__begin` Begin iterator of input sequence.
- `__end` End iterator of input sequence.
- `__comp` Comparator.

Definition at line 97 of file sort.h.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, and `_GLIBCXX_CALL`.

Here is the call graph for this function:



Choose quicksort for parallel sorting.

***\_\_begin*** Begin iterator of input sequence.  
***\_\_end*** End iterator of input sequence.  
***\_\_comp*** Comparator.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, `__parallel_sort_qs()`, and `_GLIBCXX_CALL`.

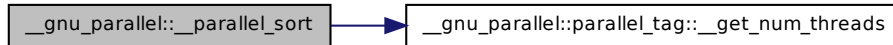
Choose multiway mergesort with exact splitting, for parallel sorting.

***\_\_begin*** Begin iterator of input sequence.  
***\_\_end*** End iterator of input sequence.  
***\_\_comp*** Comparator.

456

References \_\_gnu\_parallel::parallel\_tag::\_\_get\_num\_threads(), and \_GLIBCXX\_CALL.

Here is the call graph for this function:



**4.6.4.43** `template<typename _RAIter, typename _Compare> void  
__gnu_parallel::__parallel_sort_qs ( _RAIter __begin, _RAIter  
__end, _Compare __comp, _ThreadIndex __num_threads )`

Unbalanced quicksort main call.

#### Parameters

- \_\_begin* Begin iterator of input sequence.
- \_\_end* End iterator input sequence, ignored.
- \_\_comp* Comparator.
- \_\_num\_threads* Number of threads that are allowed to work on this part.

Definition at line 152 of file quicksort.h.

References \_\_parallel\_sort\_qs\_conquer(), and \_GLIBCXX\_CALL.

Referenced by \_\_parallel\_sort().

**4.6.4.44** `template<typename _RAIter, typename _Compare> void  
__gnu_parallel::__parallel_sort_qs_conquer ( _RAIter __begin,  
_RAIter __end, _Compare __comp, _ThreadIndex __num_threads )`

Unbalanced quicksort conquer step.

#### Parameters

- \_\_begin* Begin iterator of subsequence.

`__end` End iterator of subsequence.

`__comp` Comparator.

`__num_threads` Number of threads that are allowed to work on this part.

Definition at line 99 of file quicksort.h.

References `__parallel_sort_qs_divide()`, and `__gnu_parallel::_Settings::get()`.

Referenced by `__parallel_sort_qs()`.

```
4.6.4.45 template<typename _RAIter , typename _Compare
> std::iterator_traits<_RAIter>::difference_type
__gnu_parallel::__parallel_sort_qs_divide (_RAIter __begin,
__RAIter __end, _Compare __comp, typename std::iterator_traits<
_RAIter >::difference_type __pivot_rank, typename
std::iterator_traits< _RAIter >::difference_type __num_samples,
_ThreadIndex __num_threads)
```

Unbalanced quicksort divide step.

#### Parameters

`__begin` Begin iterator of subsequence.

`__end` End iterator of subsequence.

`__comp` Comparator.

`__pivot_rank` Desired \_\_rank of the pivot.

`__num_samples` Choose pivot from that many samples.

`__num_threads` Number of threads that are allowed to work on this part.

Definition at line 51 of file quicksort.h.

References `__parallel_partition()`, and `min()`.

Referenced by `__parallel_sort_qs_conquer()`.

```
4.6.4.46 template<typename _RAIter , typename _Compare > void
__gnu_parallel::__parallel_sort_qsb (_RAIter __begin, _RAIter
__end, _Compare __comp, _ThreadIndex __num_threads)
```

Top-level quicksort routine.

**Parameters**

- `__begin` Begin iterator of sequence.
- `__end` End iterator of sequence.
- `__comp` Comparator.
- `__num_threads` Number of threads that are allowed to work on this part.

Definition at line 430 of file `balanced_quicksort.h`.

References `__qsb_conquer()`, `__rd_log2()`, `_GLIBCXX_CALL`, and `std::make_pair()`.

Referenced by `__parallel_sort()`.

**4.6.4.47** `template<typename _Iter, class _OutputIterator> _OutputIterator  
__gnu_parallel::__parallel_unique_copy ( _Iter __first, _Iter  
__last, _OutputIterator __result ) [inline]`

Parallel `std::unique_copy()`, without explicit equality predicate.

**Parameters**

- `__first` Begin iterator of input sequence.
- `__last` End iterator of input sequence.
- `__result` Begin iterator of result \_\_sequence.

**Returns**

End iterator of result \_\_sequence.

Definition at line 186 of file `unique_copy.h`.

References `__parallel_unique_copy()`.

**4.6.4.48** `template<typename _Iter, class _OutputIterator, class  
_BinaryPredicate> _OutputIterator __gnu_parallel::__parallel_  
unique_copy ( _Iter __first, _Iter __last, _OutputIterator __result,  
_BinaryPredicate __binary_pred )`

Parallel `std::unique_copy()`, w/\_o explicit equality predicate.

**Parameters**

- `__first` Begin iterator of input sequence.

`__last` End iterator of input sequence.  
`__result` Begin iterator of result `__sequence`.  
`__binary_pred` Equality predicate.

**Returns**

End iterator of result `__sequence`.

Definition at line 50 of file `unique_copy.h`.

References `_GLIBCXX_CALL`, and `equally_split()`.

Referenced by `__parallel_unique_copy()`.

**4.6.4.49** `template<typename _RAIter, typename _Compare> void  
 __gnu_parallel::__qsb_conquer ( _QSBThreadLocal< _RAIter >  
 ** __tls, _RAIter __begin, _RAIter __end, _Compare __comp,  
 _ThreadIndex __iam, _ThreadIndex __num_threads, bool  
 __parent_wait )`

Quicksort conquer step.

**Parameters**

`__tls` Array of thread-local storages.  
`__begin` Begin iterator of subsequence.  
`__end` End iterator of subsequence.  
`__comp` Comparator.  
`__iam` Number of the thread processing this function.  
`__num_threads` Number of threads that are allowed to work on this part.

Definition at line 171 of file `balanced_quicksort.h`.

References `__qsb_divide()`, `__qsb_local_sort_with_helping()`, `__gnu_parallel::_QSBThreadLocal< _RAIter >::_M_elements_leftover`, and `__gnu_parallel::_QSBThreadLocal< _RAIter >::_M_initial`.

Referenced by `__parallel_sort_qsb()`.

**4.6.4.50** `template<typename _RAIter, typename _Compare  
 > std::iterator_traits<_RAIter>::difference_type  
 __gnu_parallel::__qsb_divide ( _RAIter __begin, _RAIter __end,  
 _Compare __comp, _ThreadIndex __num_threads )`

Balanced quicksort divide step.

#### Parameters

- `__begin` Begin iterator of subsequence.
- `__end` End iterator of subsequence.
- `__comp` Comparator.
- `__num_threads` Number of threads that are allowed to work on this part.

#### Precondition

`(__end-__begin)>=1`

Definition at line 100 of file `balanced_quicksort.h`.

References `__median_of_three_iterators()`, and `__parallel_partition()`.

Referenced by `__qsb_conquer()`.

**4.6.4.51** `template<typename _RAIter, typename _Compare> void  
__gnu_parallel::__qsb_local_sort_with_helping( _QSBThreadLocal<  
_RAIter> ** __tls, _Compare & __comp, _ThreadIndex __iam,  
bool __wait )`

Quicksort step doing load-balanced local sort.

#### Parameters

- `__tls` Array of thread-local storages.
- `__comp` Comparator.
- `__iam` Number of the thread processing this function.

Definition at line 247 of file `balanced_quicksort.h`.

References `__yield()`, `_GLIBCXX_ASSERTIONS`, `__gnu_parallel::_QSBThreadLocal< _RAIter >::_M_elements_leftover`, `__gnu_parallel::_QSBThreadLocal< _RAIter >::_M_initial`, `__gnu_parallel::_QSBThreadLocal< _RAIter >::_M_leftover_parts`, `__gnu_parallel::_QSBThreadLocal< _RAIter >::_M_num_threads`, `__gnu_parallel::_Settings::get()`, `std::make_pair()`, `__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::pop_front()`, `__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::push_front()`, and `__gnu_parallel::_Settings::sort_qsb_base_case_maximal_n`.

Referenced by `__qsb_conquer()`.



**4.6.4.52** `template<typename _RandomNumberGenerator > int  
__gnu_parallel::__random_number_pow2 ( int __logp,  
_RandomNumberGenerator & __rng ) [inline]`

Generate a random number in  $[0, 2^{\text{__logp}})$ .

#### Parameters

`__logp` Logarithm (basis 2) of the upper range `__bound`.  
`__rng` Random number generator to use.

Definition at line 115 of file `random_shuffle.h`.

Referenced by `__parallel_random_shuffle_drs_pu()`, and `__sequential_random_shuffle()`.

**4.6.4.53** `template<typename _Size > _Size __gnu_parallel::__rd_log2 ( _Size  
__n ) [inline]`

Calculates the rounded-down logarithm of `__n` for base 2.

#### Parameters

`__n` Argument.

#### Returns

Returns 0 for any argument  $< 1$ .

Definition at line 102 of file `parallel/base.h`.

Referenced by `__parallel_random_shuffle_drs()`, `__parallel_sort_qsb()`, `__round_up_to_pow2()`, `__sequential_random_shuffle()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`, `multiseq_partition()`, and `multiseq_selection()`.

**4.6.4.54** `template<typename _Tp > _Tp __gnu_parallel::__round_up_to_pow2  
( _Tp __x )`

Round up to the next greater power of 2.

#### Parameters

`__x` Integer to round up

Definition at line 246 of file `random_shuffle.h`.

References `__rd_log2()`.

Referenced by `__parallel_random_shuffle_drs()`, `__sequential_random_shuffle()`, and `multiseq_selection()`.

**4.6.4.55** `template<typename __RAIter1 , typename __RAIter2 , typename  
__Pred > __RAIter1 __gnu_parallel::__search_template ( __RAIter1  
__begin1, __RAIter1 __end1, __RAIter2 __begin2, __RAIter2  
__end2, __Pred __pred )`

Parallel `std::search`.

#### Parameters

`__begin1` Begin iterator of first sequence.  
`__end1` End iterator of first sequence.  
`__begin2` Begin iterator of second sequence.  
`__end2` End iterator of second sequence.  
`__pred` Find predicate.

#### Returns

Place of finding in first sequences.

Definition at line 81 of file `search.h`.

References `__calc_borders()`, `_GLIBCXX_CALL`, `equally_split()`, and `min()`.

**4.6.4.56** `template<bool __stable, bool __sentinels, typename __RAIterIterator ,  
typename __RAIter3 , typename __DifferenceTp , typename __Compare  
> __RAIter3 __gnu_parallel::__sequential_multiway_merge (   
__RAIterIterator __seqs_begin, __RAIterIterator __seqs_end,  
__RAIter3 __target, const typename std::iterator_traits< typename  
std::iterator_traits< __RAIterIterator >::value_type::first_type  
>::value_type & __sentinel, __DifferenceTp __length, __Compare  
__comp )`

Sequential multi-way merging switch.

The `_GLIBCXX_PARALLEL_DECISION` is based on the branching factor and run-time settings.

**Parameters**

- `__seqs_begin` Begin iterator of iterator pair input sequence.
- `__seqs_end` End iterator of iterator pair input sequence.
- `__target` Begin iterator of output sequence.
- `__comp` Comparator.
- `__length` Maximum length to merge, possibly larger than the number of elements available.
- `__stable` Stable merging incurs a performance penalty.
- `__sentinel` The sequences have `__a` `__sentinel` element.

**Returns**

End iterator of output sequence.

Definition at line 917 of file `multiway_merge.h`.

References `__is_sorted()`, `__merge_advance()`, `_GLIBCXX_CALL`, and `__GLIBCXX_PARALLEL_LENGTH`.

Referenced by `multiway_merge()`, and `multiway_merge_sentinels()`.

**4.6.4.57** `template<typename _RAIter, typename _RandomNumberGenerator>  
> void __gnu_parallel::__sequential_random_shuffle ( _RAIter  
__begin, _RAIter __end, _RandomNumberGenerator & __rng )`

Sequential cache-efficient random shuffle.

**Parameters**

- `__begin` Begin iterator of sequence.
- `__end` End iterator of sequence.
- `__rng` Random number generator to use.

Definition at line 408 of file `random_shuffle.h`.

References `__random_number_pow2()`, `__rd_log2()`, `__round_up_to_pow2()`, `__gnu_parallel::Settings::L2_cache_size`, `std::min()`, and `__gnu_parallel::Settings::TLB_size`.

Referenced by `__parallel_random_shuffle_drs()`.

**4.6.4.58** `template<typename _Iter > void __gnu_parallel::__shrink (`  
`std::vector< _Iter > & __os_starts, size_t & __count_to_two, size_t`  
`& __range_length )`

Combines two ranges into one and thus halves the number of ranges.

#### Parameters

`__os_starts` Start positions worked on (oversampled).

`__count_to_two` Counts up to 2.

`__range_length` Current length of a chunk.

Definition at line 70 of file list\_partition.h.

References `std::vector< _Tp, _Alloc >::size()`.

Referenced by `__shrink_and_double()`.

**4.6.4.59** `template<typename _Iter > void __gnu_parallel::__shrink_-`  
`and_double ( std::vector< _Iter > & __os_starts, size_t &`  
`__count_to_two, size_t & __range_length, const bool __make_twice )`

Shrinks and doubles the ranges.

#### Parameters

`__os_starts` Start positions worked on (oversampled).

`__count_to_two` Counts up to 2.

`__range_length` Current length of a chunk.

`__make_twice` Whether the `__os_starts` is allowed to be grown or not

Definition at line 50 of file list\_partition.h.

References `__shrink()`, `std::vector< _Tp, _Alloc >::resize()`, and `std::vector< _Tp, _Alloc >::size()`.

Referenced by `list_partition()`.

**4.6.4.60** `void __gnu_parallel::__yield ( ) [inline]`

Yield the control to another thread, without waiting for the end to the time slice.

Definition at line 354 of file `parallel/compatibility.h`.

Referenced by `__for_each_template_random_access_workstealing()`, and `__qsb_local_sort_with_helping()`.

**4.6.4.61** `template<typename _DifferenceType , typename _OutputIterator >  
_OutputIterator __gnu_parallel::equally_split ( _DifferenceType __n,  
_ThreadIndex __num_threads, _OutputIterator __s )`

function to split a sequence into parts of almost equal size.

The resulting sequence `__s` of length `__num_threads+1` contains the splitting positions when splitting the range `[0,__n)` into parts of almost equal size (plus minus 1). The first entry is 0, the last one `n`. There may result empty parts.

#### Parameters

`__n` Number of elements  
`__num_threads` Number of parts  
`__s` Splitters

#### Returns

End of `__splitter` sequence, i.e. `__s+__num_threads+1`

Definition at line 48 of file `equally_split.h`.

Referenced by `__determine_samples()`, `__find_template()`, `__parallel_partial_sum_linear()`, `__parallel_unique_copy()`, `__search_template()`, and `multiway_merge_exact_splitting()`.

**4.6.4.62** `template<typename _DifferenceType > _DifferenceType  
__gnu_parallel::equally_split_point ( _DifferenceType __n,  
_ThreadIndex __num_threads, _ThreadIndex __thread_no )`

function to split a sequence into parts of almost equal size.

Returns the position of the splitting point between thread number `__thread_no` (included) and thread number `__thread_no+1` (excluded).

#### Parameters

`__n` Number of elements  
`__num_threads` Number of parts

**Returns**

splitting point

Definition at line 74 of file `equally_split.h`.

Referenced by `__for_each_template_random_access_ed()`.

**4.6.4.63** `template<typename _Iter, typename _FunctorType> size_t  
__gnu_parallel::list_partition ( const _Iter __begin, const _Iter  
__end, _Iter * __starts, size_t * __lengths, const int __num_parts,  
_FunctorType & __f, int __oversampling = 0 )`

Splits a sequence given by input iterators into parts of almost equal size.

The function needs only one pass over the sequence.

**Parameters**

`__begin` Begin iterator of input sequence.

`__end` End iterator of input sequence.

`__starts` Start iterators for the resulting parts, dimension `__num_parts+1`. For convenience, `__starts [ __num_parts ]` contains the end iterator of the sequence.

`__lengths` Length of the resulting parts.

`__num_parts` Number of parts to split the sequence into.

`__f` Functor to be applied to each element by traversing `__it`

`__oversampling` Oversampling factor. If 0, then the partitions will differ in at most  $\{ \{ \text{__end} \} - \{ \text{__begin} \} \}$  `__elements`. Otherwise, the ratio between the longest and the shortest part is bounded by  $1 / ( \{ \text{__oversampling} \} \{ \text{num} \} )$

**Returns**

Length of the whole sequence.

Definition at line 101 of file `list_partition.h`.

References `__shrink_and_double()`, and `std::vector<_Tp, _Alloc>::size()`.

**4.6.4.64** `template<typename _Tp> const _Tp& __gnu_parallel::max ( const  
_Tp & __a, const _Tp & __b ) [inline]`

Equivalent to `std::max`.

Definition at line 150 of file `parallel/base.h`.

Referenced by `__parallel_nth_element()`, `multiseq_partition()`, and `multiseq_selection()`.

**4.6.4.65** `template<typename _Tp> const _Tp& __gnu_parallel::min ( const  
_Tp & __a, const _Tp & __b ) [inline]`

Equivalent to `std::min`.

Definition at line 144 of file `parallel/base.h`.

Referenced by `__for_each_template_random_access_workstealing()`, `__parallel_sort_qs_divide()`, `__search_template()`, `multiseq_partition()`, and `multiseq_selection()`.

**4.6.4.66** `template<typename _RanSeqs, typename _RankType,  
typename _RankIterator, typename _Compare> void __gnu_  
parallel::multiseq_partition ( _RanSeqs __begin_seqs, _RanSeqs  
__end_seqs, _RankType __rank, _RankIterator __begin_offsets,  
_Compare __comp = std::less< typename std::iterator_  
traits<typename std::iterator_traits<_  
_RanSeqs>::value_type:: first_type>::value_type> ()  
)`

Splits several sorted sequences at a certain global `__rank`, resulting in a splitting point for each sequence. The sequences are passed via a sequence of random-access iterator pairs, none of the sequences may be empty. If there are several equal elements across the split, the ones on the `__left` side will be chosen from sequences with smaller number.

#### Parameters

- `__begin_seqs` Begin of the sequence of iterator pairs.
- `__end_seqs` End of the sequence of iterator pairs.
- `__rank` The global rank to partition at.
- `__begin_offsets` A random-access `__sequence` `__begin` where the `__result` will be stored in. Each element of the sequence is an iterator that points to the first element on the greater part of the respective `__sequence`.
- `__comp` The ordering functor, defaults to `std::less<_Tp>`.

Definition at line 124 of file `multiseq_selection.h`.

References `__rd_log2()`, `_GLIBCXX_CALL`, `std::vector< _Tp, _Alloc >::begin()`, `std::distance()`, `__gnu_cxx::distance()`, `std::priority_queue< _Tp, _Sequence, _Compare >::empty()`, `std::vector< _Tp, _Alloc >::end()`, `std::make_pair()`, `max()`, `min()`, `std::priority_queue< _Tp, _Sequence, _Compare >::pop()`, `std::priority_queue< _Tp, _Sequence, _Compare >::push()`, `std::vector< _Tp, _Alloc >::push_back()`, and `std::priority_queue< _Tp, _Sequence, _Compare >::top()`.

Referenced by `multiway_merge_exact_splitting()`.

**4.6.4.67** `template<typename _Tp, typename _RanSeqs, typename _RankType, typename _Compare> _Tp __gnu_parallel::multiseq_selection ( _RanSeqs begin_seqs, _RanSeqs end_seqs, _RankType rank, _RankType & offset, _Compare comp = std::less<_Tp> () )`

Selects the element at a certain global `__rank` from several sorted sequences.

The sequences are passed via a sequence of random-access iterator pairs, none of the sequences may be empty.

#### Parameters

- `begin_seqs` Begin of the sequence of iterator pairs.
- `end_seqs` End of the sequence of iterator pairs.
- `rank` The global rank to partition at.
- `offset` The rank of the selected element in the global subsequence of elements equal to the selected element. If the selected element is unique, this number is 0.
- `comp` The ordering functor, defaults to `std::less`.

Definition at line 390 of file `multiseq_selection.h`.

References `__rd_log2()`, `__round_up_to_pow2()`, `_GLIBCXX_CALL`, `std::vector< _Tp, _Alloc >::begin()`, `std::distance()`, `__gnu_cxx::distance()`, `std::priority_queue< _Tp, _Sequence, _Compare >::empty()`, `std::vector< _Tp, _Alloc >::end()`, `std::make_pair()`, `max()`, `min()`, `std::priority_queue< _Tp, _Sequence, _Compare >::pop()`, `std::priority_queue< _Tp, _Sequence, _Compare >::push()`, `std::vector< _Tp, _Alloc >::push_back()`, and `std::priority_queue< _Tp, _Sequence, _Compare >::top()`.



**4.6.4.68** `template<typename _RAIterPairIterator, typename _RAIterOut,
typename _DifferenceTp, typename _Compare> _RAIterOut
__gnu_parallel::multiway_merge ( _RAIterPairIterator
__seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut
__target, _DifferenceTp __length, _Compare __comp,
__gnu_parallel::sequential_tag )`

Multiway Merge Frontend.

Merge the sequences specified by `seqs_begin` and `__seqs_end` into `__target`. `__seqs_begin` and `__seqs_end` must point to a sequence of pairs. These pairs must contain an iterator to the beginning of a sequence in their first entry and an iterator the `_M_end` of the same sequence in their second entry.

Ties are broken arbitrarily. See `stable_multiway_merge` for a variant that breaks ties by sequence number but is slower.

The first entries of the pairs (i.e. the begin iterators) will be moved forward.

The output sequence has to provide enough space for all elements that are written to it.

This function will merge the input sequences:

- not stable
- parallel, depending on the input size and Settings
- using sampling for splitting
- not using sentinels

Example:

```
int sequences[10][10];
for (int __i = 0; __i < 10; ++__i)
 for (int __j = 0; __j < 10; ++__j)
 sequences[__i][__j] = __j;

int __out[33];
std::vector<std::pair<int*>> > seqs;
for (int __i = 0; __i < 10; ++__i)
 { seqs.push(std::make_pair<int*>(sequences[__i],
 sequences[__i] + 10)) }

multiway_merge(seqs.begin(), seqs.end(), __target, std::less<int>(), 33);
```

**See also**

`stable_multiway_merge`

**Precondition**

All input sequences must be sorted.

Target must provide enough space to merge out length elements or the number of elements in all sequences, whichever is smaller.

**Postcondition**

[`__target`, return `__value`) contains merged `__elements` from the input sequences.  
return `__value` - `__target` = min(`__length`, number of elements in all sequences).

**Parameters**

`_RAIterPairIterator` iterator over sequence of pairs of iterators

`_RAIterOut` iterator over target sequence

`_DifferenceTp` difference type for the sequence

`_Compare` strict weak ordering type to compare elements in sequences

`__seqs_begin` `__begin` of sequence `__sequence`

`__seqs_end` `_M_end` of sequence `__sequence`

`__target` target sequence to merge to.

`__comp` strict weak ordering to use for element comparison.

`__length` Maximum length to merge, possibly larger than the number of elements available.

**Returns**

`_M_end` iterator of output sequence

Definition at line 1410 of file `multiway_merge.h`.

References `__sequential_multiway_merge()`, and `_GLIBCXX_CALL`.

```
4.6.4.69 template<template< typename RAI, typename C > class
 iterator, typename _RAIterIterator , typename _RAIter3 ,
 typename _DifferenceTp , typename _Compare > _RAIter3
 __gnu_parallel::multiway_merge_3_variant (_RAIterIterator
 __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target,
 _DifferenceTp __length, _Compare __comp)
```

Highly efficient 3-way merging procedure.

Merging is done with the algorithm implementation described by Peter Sanders. Basically, the idea is to minimize the number of necessary comparison after merging an element. The implementation trick that makes this fast is that the order of the sequences is stored in the instruction pointer (translated into labels in C++).

This works well for merging up to 4 sequences.

Note that making the merging stable does *not* come at a performance hit.

Whether the merging is done guarded or unguarded is selected by the used iterator class.

#### Parameters

- `__seqs_begin` Begin iterator of iterator pair input sequence.
- `__seqs_end` End iterator of iterator pair input sequence.
- `__target` Begin iterator of output sequence.
- `__comp` Comparator.
- `__length` Maximum length to merge, less equal than the total number of elements available.

#### Returns

End iterator of output sequence.

Definition at line 234 of file `multiway_merge.h`.

References `_GLIBCXX_CALL`.

**4.6.4.70** `template<template< typename RAI, typename C > class  
iterator, typename _RAIterIterator , typename _RAIter3 ,  
typename _DifferenceTp , typename _Compare > _RAIter3  
__gnu_parallel::multiway_merge_4_variant ( _RAIterIterator  
__seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target,  
_DifferenceTp __length, _Compare __comp )`

Highly efficient 4-way merging procedure.

Merging is done with the algorithm implementation described by Peter Sanders. Basically, the idea is to minimize the number of necessary comparison after merging an element. The implementation trick that makes this fast is that the order of the sequences is stored in the instruction pointer (translated into goto labels in C++).

This works well for merging up to 4 sequences.

Note that making the merging stable does *not* come at a performance hit.

Whether the merging is done guarded or unguarded is selected by the used iterator class.

**Parameters**

- `__seqs_begin` Begin iterator of iterator pair input sequence.
- `__seqs_end` End iterator of iterator pair input sequence.
- `__target` Begin iterator of output sequence.
- `__comp` Comparator.
- `__length` Maximum length to merge, less equal than the total number of elements available.

**Returns**

End iterator of output sequence.

Definition at line 353 of file `multiway_merge.h`.

References `_GLIBCXX_CALL`.

```
4.6.4.71 template<bool __stable, typename _RAIterIterator ,
 typename _Compare , typename _DifferenceType > void
 __gnu_parallel::multiway_merge_exact_splitting (_RAIterIterator
 __seqs_begin, _RAIterIterator __seqs_end, _DifferenceType
 __length, _DifferenceType __total_length, _Compare __comp,
 std::vector< std::pair< _DifferenceType, _DifferenceType > > *
 __pieces)
```

Exact splitting for parallel multiway-merge routine.

None of the passed sequences may be empty.

Definition at line 1112 of file `multiway_merge.h`.

References `_GLIBCXX_PARALLEL_LENGTH`, `std::vector< _Tp, _Alloc >::begin()`, `std::vector< _Tp, _Alloc >::end()`, `equally_split()`, `multiseq_partition()`, and `std::vector< _Tp, _Alloc >::resize()`.

Referenced by `__parallel_merge_advance()`.

```
4.6.4.72 template<typename _LT , typename _RAIterIterator , typename
 _RAIter3 , typename _DifferenceTp , typename _Compare > _RAIter3
 __gnu_parallel::multiway_merge_loser_tree (_RAIterIterator
 __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target,
 _DifferenceTp __length, _Compare __comp)
```

Multi-way merging procedure for a high branching factor, guarded case.

This merging variant uses a `LoserTree` class as selected by `__LT`.

Stability is selected through the used `LoserTree` class `__LT`.

At least one non-empty sequence is required.

#### Parameters

`__seqs_begin` Begin iterator of iterator pair input sequence.

`__seqs_end` End iterator of iterator pair input sequence.

`__target` Begin iterator of output sequence.

`__comp` Comparator.

`__length` Maximum length to merge, less equal than the total number of elements available.

#### Returns

End iterator of output sequence.

Definition at line 484 of file `multiway_merge.h`.

References `_GLIBCXX_CALL`, and `_GLIBCXX_PARALLEL_LENGTH`.

**4.6.4.73** `template<typename UnguardedLoserTree, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare > _RAIter3 __gnu_parallel::multiway_merge_loser_tree_sentinel ( _RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp )`

Multi-way merging procedure for a high branching factor, requiring sentinels to exist.

#### Parameters

`__stable` The value must be the same as for the used `LoserTrees`.

***UnguardedLoserTree*** `_LoserTree` variant to use for the unguarded merging.

***GuardedLoserTree*** `_LoserTree` variant to use for the guarded merging.

`__seqs_begin` Begin iterator of iterator pair input sequence.

`__seqs_end` End iterator of iterator pair input sequence.

`__target` Begin iterator of output sequence.

`__comp` Comparator.

`__length` Maximum length to merge, less equal than the total number of elements available.

### Returns

End iterator of output sequence.

Definition at line 658 of file `multiway_merge.h`.

References `__is_sorted()`, and `_GLIBCXX_CALL`.

**4.6.4.74** `template<typename _LT , typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Compare > _RAIter3 __gnu_parallel::multiway_merge_loser_tree_unguarded ( _RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp )`

Multi-way merging procedure for a high branching factor, unguarded case.

Merging is done using the `LoserTree` class `_LT`.

Stability is selected by the used `LoserTrees`.

### Precondition

No input will run out of elements during the merge.

### Parameters

`__seqs_begin` Begin iterator of iterator pair input sequence.

`__seqs_end` End iterator of iterator pair input sequence.

`__target` Begin iterator of output sequence.

`__comp` Comparator.

`__length` Maximum length to merge, less equal than the total number of elements available.

### Returns

End iterator of output sequence.

Definition at line 567 of file `multiway_merge.h`.

References `_GLIBCXX_CALL`.

**4.6.4.75** `template<bool __stable, typename _RAIterIterator ,  
 typename _Compare , typename _DifferenceType > void  
 __gnu_parallel::multiway_merge_sampling_splitting (   
 _RAIterIterator __seqs_begin, _RAIterIterator __seqs_end,  
 _DifferenceType __length, _DifferenceType __total_length,  
 _Compare __comp, std::vector< std::pair< _DifferenceType,  
 _DifferenceType > > * __pieces )`

Sampling based splitting for parallel multiway-merge routine.

Definition at line 1032 of file `multiway_merge.h`.

References `_GLIBCXX_PARALLEL_LENGTH`, `__gnu_parallel::_Settings::get()`,  
 and `__gnu_parallel::_Settings::merge_oversampling`.

**4.6.4.76** `template<typename _RAIterPairIterator , typename _RAIterOut ,  
 typename _DifferenceTp , typename _Compare > _RAIterOut  
 __gnu_parallel::multiway_merge_sentinels ( _RAIterPairIterator  
 __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut  
 __target, _DifferenceTp __length, _Compare __comp,  
 __gnu_parallel::sequential_tag )`

Multiway Merge Frontend.

Merge the sequences specified by `seqs_begin` and `__seqs_end` into `__target`. `__seqs_begin` and `__seqs_end` must point to a sequence of pairs. These pairs must contain an iterator to the beginning of a sequence in their first entry and an iterator the `_M_end` of the same sequence in their second entry.

Ties are broken arbitrarily. See `stable_multiway_merge` for a variant that breaks ties by sequence number but is slower.

The first entries of the pairs (i.e. the begin iterators) will be moved forward accordingly.

The output sequence has to provide enough space for all elements that are written to it.

This function will merge the input sequences:

- not stable
- parallel, depending on the input size and Settings
- using sampling for splitting
- using sentinels

You have to take care that the element the `_M_end` iterator points to is readable and contains a value that is greater than any other non-sentinel value in all sequences.

Example:

```
int sequences[10][11];
for (int __i = 0; __i < 10; ++__i)
 for (int __j = 0; __i < 11; ++__j)
 sequences[__i][__j] = __j; // __last one is sentinel!

int __out[33];
std::vector<std::pair<int*> > seqs;
for (int __i = 0; __i < 10; ++__i)
 { seqs.push(std::make_pair<int*>(sequences[__i],
 sequences[__i] + 10)) }

multiway_merge(seqs.begin(), seqs.end(), __target, std::less<int>(), 33);
```

### Precondition

All input sequences must be sorted.

Target must provide enough space to merge out length elements or the number of elements in all sequences, whichever is smaller.

For each `__i`, `__seqs_begin[__i].second` must be the end marker of the sequence, but also reference the one more `__sentinel` element.

### Postcondition

`[__target, return __value)` contains merged `__elements` from the input sequences.  
`return __value - __target = min(__length, number of elements in all sequences).`

### See also

`stable_multiway_merge_sentinels`

### Parameters

**`_RAIterPairIterator`** iterator over sequence of pairs of iterators

**`_RAIterOut`** iterator over target sequence

**`_DifferenceTp`** difference type for the sequence

**`_Compare`** strict weak ordering type to compare elements in sequences

**`__seqs_begin`** `__begin` of sequence `__sequence`

**`__seqs_end`** `_M_end` of sequence `__sequence`

**`__target`** target sequence to merge to.

**`__comp`** strict weak ordering to use for element comparison.



`__length` Maximum length to merge, possibly larger than the number of elements available.

### Returns

`_M_end` iterator of output sequence

Definition at line 1774 of file `multiway_merge.h`.

References `__sequential_multiway_merge()`, and `_GLIBCXX_CALL`.

```
4.6.4.77 template<bool __stable, bool __sentinels, typename
 _RAIterIterator, typename _RAIter3, typename _DifferenceTp,
 typename _Splitter, typename _Compare > _RAIter3
 __gnu_parallel::parallel_multiway_merge (_RAIterIterator
 __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target,
 _Splitter __splitter, _DifferenceTp __length, _Compare __comp,
 _ThreadIndex __num_threads)
```

Parallel multi-way merge routine.

The `_GLIBCXX_PARALLEL_DECISION` is based on the branching factor and run-time settings.

Must not be called if the number of sequences is 1.

### Parameters

`__Splitter` functor to split input (either `__exact` or sampling based)

`__seqs_begin` Begin iterator of iterator pair input sequence.

`__seqs_end` End iterator of iterator pair input sequence.

`__target` Begin iterator of output sequence.

`__comp` Comparator.

`__length` Maximum length to merge, possibly larger than the number of elements available.

`__stable` Stable merging incurs a performance penalty.

`__sentinel` Ignored.

### Returns

End iterator of output sequence.

Definition at line 1217 of file `multiway_merge.h`.

References `__is_sorted()`, `_GLIBCXX_CALL`, `_GLIBCXX_PARALLEL_LENGTH`, `__gnu_parallel::Settings::get()`, `std::make_pair()`, and `__gnu_parallel::Settings::merge_oversampling`.

Referenced by `__parallel_merge_advance()`.

**4.6.4.78** `template<bool __stable, bool __exact, typename _RAIter, typename _Compare> void __gnu_parallel::parallel_sort_mwms ( _RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads )`

PMWMS main call.

#### Parameters

- `__begin` Begin iterator of sequence.
- `__end` End iterator of sequence.
- `__comp` Comparator.
- `__n` Length of sequence.
- `__num_threads` Number of threads to use.

Definition at line 394 of file `multiway_mergesort.h`.

References `_GLIBCXX_CALL`, `__gnu_parallel::PMWMSortingData< _RAIter >::M_num_threads`, `__gnu_parallel::PMWMSortingData< _RAIter >::M_offsets`, `__gnu_parallel::PMWMSortingData< _RAIter >::M_pieces`, `__gnu_parallel::PMWMSortingData< _RAIter >::M_samples`, `__gnu_parallel::PMWMSortingData< _RAIter >::M_source`, `__gnu_parallel::PMWMSortingData< _RAIter >::M_starts`, `__gnu_parallel::PMWMSortingData< _RAIter >::M_temporary`, `__gnu_parallel::Settings::get()`, `std::vector< _Tp, _Alloc >::resize()`, and `__gnu_parallel::Settings::sort_mwms_oversampling`.

**4.6.4.79** `template<bool __stable, bool __exact, typename _RAIter, typename _Compare> void __gnu_parallel::parallel_sort_mwms_pu ( _PMWMSortingData< _RAIter > * __sd, _Compare & __comp )`

PMWMS code executed by each thread.

#### Parameters

- `__sd` Pointer to algorithm data.

`__comp` Comparator.

Definition at line 308 of file `multiway_mergesort.h`.

References `__gnu_parallel::PMWMSortingData< _RAIter >::_M_num_threads`, `__gnu_parallel::PMWMSortingData< _RAIter >::_M_pieces`, `__gnu_parallel::PMWMSortingData< _RAIter >::_M_source`, `__gnu_parallel::PMWMSortingData< _RAIter >::_M_starts`, `__gnu_parallel::PMWMSortingData< _RAIter >::_M_temporary`, `__gnu_parallel::Settings::get()`, `std::make_pair()`, `__gnu_parallel::Settings::sort_mwms_oversampling`, and `std::uninitialized_copy()`.

#### 4.6.5 Variable Documentation

##### 4.6.5.1 `const int __gnu_parallel::_CASable_bits` `[static]`

Number of bits of `_CASable`.

Definition at line 130 of file `types.h`.

Referenced by `__decode2()`, and `__encode2()`.

##### 4.6.5.2 `const _CASable __gnu_parallel::_CASable_mask` `[static]`

`_CASable` with the right half of bits set to 1.

Definition at line 133 of file `types.h`.

Referenced by `__decode2()`.

## 4.7 `__gnu_pbds` Namespace Reference

GNU extensions for policy-based data structures for public use.

### Classes

- struct `associative_container_tag`  
*Basic associative-container.*
- class `basic_hash_table`  
*An abstract basic hash-based associative container.*

- struct `basic_hash_tag`  
*Basic hash.*
- class `basic_tree`  
*An abstract basic tree-like (tree, trie) associative container.*
- struct `basic_tree_tag`  
*Basic tree.*
- struct `binary_heap_tag`  
*Binary-heap (array-based).*
- struct `binomial_heap_tag`  
*Binomial-heap.*
- class `cc_hash_table`  
*A concrete collision-chaining hash-based associative container.*
- struct `cc_hash_tag`  
*Collision-chaining hash.*
- class `container_base`  
*An abstract basic associative container.*
- struct `container_tag`  
*Base data structure tag.*
- struct `container_traits`  
*container\_traits*
- class `gp_hash_table`  
*A concrete general-probing hash-based associative container.*
- struct `gp_hash_tag`  
*General-probing hash.*
- class `list_update`  
*A list-update based associative container.*
- struct `list_update_tag`  
*List-update.*

- struct [null\\_mapped\\_type](#)  
*A mapped-policy indicating that an associative container is a set.*
- struct [ov\\_tree\\_tag](#)  
*Ordered-vector tree.*
- struct [pairing\\_heap\\_tag](#)  
*Pairing-heap.*
- struct [pat\\_trie\\_tag](#)  
*PATRICIA trie.*
- struct [priority\\_queue\\_tag](#)  
*Basic priority-queue.*
- struct [rb\\_tree\\_tag](#)  
*Red-black tree.*
- struct [rc\\_binomial\\_heap\\_tag](#)  
*Redundant-counter binomial-heap.*
- struct [sequence\\_tag](#)  
*Basic sequence.*
- struct [splay\\_tree\\_tag](#)  
*Splay tree.*
- struct [string\\_tag](#)  
*Basic string container; inclusive of strings, ropes, etc.*
- struct [thin\\_heap\\_tag](#)  
*Thin heap.*
- class [tree](#)  
*A concrete basic tree-based associative container.*
- struct [tree\\_tag](#)  
*tree.*
- class [trie](#)  
*A concrete basic trie-based associative container.*

- struct [trie\\_tag](#)  
*trie.*

### Typedefs

- typedef void **trivial\_iterator\_difference\_type**

### Functions

- void **\_\_throw\_container\_error** (void)
- void **\_\_throw\_insert\_error** (void)
- void **\_\_throw\_join\_error** (void)
- void **\_\_throw\_resize\_error** (void)

#### 4.7.1 Detailed Description

GNU extensions for policy-based data structures for public use.

## 4.8 `__gnu_profile` Namespace Reference

GNU profile code for public use.

### Classes

- class [\\_\\_container\\_size\\_info](#)  
*A container size instrumentation line in the object table.*
- class [\\_\\_container\\_size\\_stack\\_info](#)  
*A container size instrumentation line in the stack table.*
- class [\\_\\_hashfunc\\_info](#)  
*A hash performance instrumentation line in the object table.*
- class [\\_\\_hashfunc\\_stack\\_info](#)  
*A hash performance instrumentation line in the stack table.*
- class [\\_\\_list2vector\\_info](#)  
*A list-to-vector instrumentation line in the object table.*
- class [\\_\\_map2umap\\_info](#)

*A map-to-unordered\_map instrumentation line in the object table.*

- class `__map2umap_stack_info`  
*A map-to-unordered\_map instrumentation line in the stack table.*
- class `__object_info_base`  
*Base class for a line in the object table.*
- struct `__reentrance_guard`  
*Reentrance guard.*
- class `__stack_hash`  
*Hash function for summary trace using call stack as index.*
- class `__stack_info_base`  
*Base class for a line in the stack table.*
- class `__trace_base`  
*Base class for all trace producers.*
- class `__trace_container_size`  
*Container size instrumentation trace producer.*
- class `__trace_hash_func`  
*Hash performance instrumentation producer.*
- class `__trace_hashtable_size`  
*Hashtable size instrumentation trace producer.*
- class `__trace_map2umap`  
*Map-to-unordered\_map instrumentation producer.*
- class `__trace_vector_size`  
*Hashtable size instrumentation trace producer.*
- class `__trace_vector_to_list`  
*Vector-to-list instrumentation producer.*
- class `__vector2list_info`  
*A vector-to-list instrumentation line in the object table.*
- class `__vector2list_stack_info`

*A vector-to-list instrumentation line in the stack table.*

- struct `__warning_data`  
*Representation of a warning.*

### Typedefs

- typedef std::vector< \_\_cost\_factor \* > `__cost_factor_vector`
- typedef std::unordered\_map< std::string, std::string > `__env_t`
- typedef void \* `__instruction_address_t`
- typedef const void \* `__object_t`
- typedef std::vector< \_\_instruction\_address\_t > `__stack_npt`
- typedef \_\_stack\_npt \* `__stack_t`
- typedef std::vector< `__warning_data` > `__warning_vector_t`

### Enumerations

- enum `__state_type` { `__ON`, `__OFF`, `__INVALID` }

### Functions

- std::size\_t `__env_to_size_t` (const char \* \_\_env\_var, std::size\_t \_\_default\_value)
- template<typename \_InputIterator, typename \_Function >  
\_Function `__for_each` (\_InputIterator \_\_first, \_InputIterator \_\_last, \_Function \_\_f)
- \_\_cost\_factor\_vector \* & `__get__cost_factors` ()
- `__env_t` & `__get__env` ()
- \_\_gnu\_cxx::\_\_mutex & `__get__global_lock` ()
- \_\_cost\_factor & `__get__list_iterate_cost_factor` ()
- \_\_cost\_factor & `__get__list_resize_cost_factor` ()
- \_\_cost\_factor & `__get__list_shift_cost_factor` ()
- \_\_cost\_factor & `__get__map_erase_cost_factor` ()
- \_\_cost\_factor & `__get__map_find_cost_factor` ()
- \_\_cost\_factor & `__get__map_insert_cost_factor` ()
- \_\_cost\_factor & `__get__map_iterate_cost_factor` ()
- \_\_cost\_factor & `__get__umap_erase_cost_factor` ()
- \_\_cost\_factor & `__get__umap_find_cost_factor` ()
- \_\_cost\_factor & `__get__umap_insert_cost_factor` ()
- \_\_cost\_factor & `__get__umap_iterate_cost_factor` ()
- \_\_cost\_factor & `__get__vector_iterate_cost_factor` ()



- `__cost_factor & __get__vector_resize_cost_factor ()`
- `__cost_factor & __get__vector_shift_cost_factor ()`
- `__trace_hash_func *& __get__S_hash_func ()`
- `__trace_hashtable_size *& __get__S_hashtable_size ()`
- `__trace_list_to_slist *& __get__S_list_to_slist ()`
- `__trace_list_to_vector *& __get__S_list_to_vector ()`
- `__trace_map2umap *& __get__S_map2umap ()`
- `std::size_t & __get__S_max_mem ()`
- `std::size_t & __get__S_max_stack_depth ()`
- `std::size_t & __get__S_max_warn_count ()`
- `const char *& __get__S_trace_file_name ()`
- `__trace_vector_size *& __get__S_vector_size ()`
- `__trace_vector_to_list *& __get__S_vector_to_list ()`
- `__stack_t __get_stack ()`
- `template<typename _Container >`  
`void __insert_top_n (_Container &__output, const typename _-`  
`Container::value_type &__value, typename _Container::size_type __n)`
- `bool __is_invalid ()`
- `bool __is_off ()`
- `bool __is_on ()`
- `int __log2 (std::size_t __size)`
- `int __log_magnitude (float __f)`
- `float __map_erase_cost (std::size_t __size)`
- `float __map_find_cost (std::size_t __size)`
- `float __map_insert_cost (std::size_t __size)`
- `std::size_t __max_mem ()`
- `FILE * __open_output_file (const char *__extension)`
- `bool __profcxx_init ()`
- `void __profcxx_init_unconditional ()`
- `void __read_cost_factors ()`
- `template<typename _ForwardIterator, typename _Tp >`  
`_ForwardIterator __remove (_ForwardIterator __first, _ForwardIterator __last,`  
`const _Tp &__value)`
- `void __report (void)`
- `void __set_cost_factors ()`
- `void __set_max_mem ()`
- `void __set_max_stack_trace_depth ()`
- `void __set_max_warn_count ()`
- `void __set_trace_path ()`
- `std::size_t __size (__stack_t __stack)`
- `std::size_t __stack_max_depth ()`

- `template<typename _Container >`  
`void __top_n (const _Container &__input, _Container &__output, typename _Container::size_type __n)`
- `void __trace_hash_func_construct (const void *)`
- `void __trace_hash_func_destruct (const void *, std::size_t, std::size_t, std::size_t)`
- `void __trace_hash_func_init ()`
- `void __trace_hash_func_report (FILE *__f, __warning_vector_t &__warnings)`
- `void __trace_hashtable_size_construct (const void *, std::size_t)`
- `void __trace_hashtable_size_destruct (const void *, std::size_t, std::size_t)`
- `void __trace_hashtable_size_init ()`
- `void __trace_hashtable_size_report (FILE *__f, __warning_vector_t &__warnings)`
- `void __trace_hashtable_size_resize (const void *, std::size_t, std::size_t)`
- `void __trace_list_to_set_construct (const void *)`
- `void __trace_list_to_set_destruct (const void *)`
- `void __trace_list_to_set_find (const void *, std::size_t)`
- `void __trace_list_to_set_insert (const void *, std::size_t, std::size_t)`
- `void __trace_list_to_set_invalid_operator (const void *)`
- `void __trace_list_to_set_iterate (const void *, std::size_t)`
- `void __trace_list_to_slist_construct (const void *)`
- `void __trace_list_to_slist_destruct (const void *)`
- `void __trace_list_to_slist_init ()`
- `void __trace_list_to_slist_operation (const void *)`
- `void __trace_list_to_slist_report (FILE *__f, __warning_vector_t &__warnings)`
- `void __trace_list_to_slist_rewind (const void *)`
- `void __trace_list_to_vector_construct (const void *)`
- `void __trace_list_to_vector_destruct (const void *)`
- `void __trace_list_to_vector_init ()`
- `void __trace_list_to_vector_insert (const void *, std::size_t, std::size_t)`
- `void __trace_list_to_vector_invalid_operator (const void *)`
- `void __trace_list_to_vector_iterate (const void *, std::size_t)`
- `void __trace_list_to_vector_report (FILE *__f, __warning_vector_t &__warnings)`
- `void __trace_list_to_vector_resize (const void *, std::size_t, std::size_t)`
- `void __trace_map_to_unordered_map_construct (const void *)`
- `void __trace_map_to_unordered_map_destruct (const void *)`
- `void __trace_map_to_unordered_map_erase (const void *, std::size_t, std::size_t)`
- `void __trace_map_to_unordered_map_find (const void *, std::size_t)`
- `void __trace_map_to_unordered_map_init ()`

- void `__trace_map_to_unordered_map_insert` (const void \*, std::size\_t, std::size\_t)
- void `__trace_map_to_unordered_map_invalidate` (const void \*)
- void `__trace_map_to_unordered_map_iterate` (const void \*, std::size\_t)
- void `__trace_map_to_unordered_map_report` (FILE \* \_\_f, \_\_warning\_vector\_t & \_\_warnings)
- void `__trace_vector_size_construct` (const void \*, std::size\_t)
- void `__trace_vector_size_destruct` (const void \*, std::size\_t, std::size\_t)
- void `__trace_vector_size_init` ()
- void `__trace_vector_size_report` (FILE \*, \_\_warning\_vector\_t &)
- void `__trace_vector_size_resize` (const void \*, std::size\_t, std::size\_t)
- void `__trace_vector_to_list_construct` (const void \*)
- void `__trace_vector_to_list_destruct` (const void \*)
- void `__trace_vector_to_list_find` (const void \*, std::size\_t)
- void `__trace_vector_to_list_init` ()
- void `__trace_vector_to_list_insert` (const void \*, std::size\_t, std::size\_t)
- void `__trace_vector_to_list_invalid_operator` (const void \*)
- void `__trace_vector_to_list_iterate` (const void \*, std::size\_t)
- void `__trace_vector_to_list_report` (FILE \*, \_\_warning\_vector\_t &)
- void `__trace_vector_to_list_resize` (const void \*, std::size\_t, std::size\_t)
- bool `__turn` (\_\_state\_type \_\_s)
- bool `__turn_off` ()
- bool `__turn_on` ()
- void `__write` (FILE \* \_\_f, \_\_stack\_t \_\_stack)
- void `__write_cost_factors` ()
- `_GLIBCXX_PROFILE_DEFINE_DATA` (\_\_state\_type, \_\_state, \_\_-INVALID)

#### 4.8.1 Detailed Description

GNU profile code for public use.

#### 4.8.2 Typedef Documentation

##### 4.8.2.1 `typedef std:: ::unordered_map<std::string, std::string> __gnu_profile::__env_t`

Internal environment. Values can be set one of two ways: 1. In config file "var = value". The default config file path is libstdcxx-profile.conf. 2. By setting process environment variables. For instance, in a Bash shell you can set the unit cost of iterating through a

map like this: `export __map_iterate_cost_factor=5.0`. If a value is set both in the input file and through an environment variable, the environment value takes precedence.

Definition at line 72 of file `profiler_trace.h`.

### 4.8.3 Function Documentation

#### 4.8.3.1 `__gnu_cxx::__mutex& __gnu_profile::__get__global_lock ( )` `[inline]`

Master lock.

Definition at line 77 of file `profiler_trace.h`.

#### 4.8.3.2 `bool __gnu_profile::__profcxx_init ( )` `[inline]`

This function must be called by each instrumentation point.

The common path is inlined fully.

Definition at line 656 of file `profiler_trace.h`.

#### 4.8.3.3 `void __gnu_profile::__report ( void )` `[inline]`

Final report method, registered with `atexit`.

This can also be called directly by user code, including signal handlers. It is protected against deadlocks by the reentrance guard in [profiler.h](#). However, when called from a signal handler that triggers while within `__gnu_profile` (under the guarded zone), no output will be produced.

Definition at line 447 of file `profiler_trace.h`.

References `std::min()`.

## 4.9 `__gnu_sequential` Namespace Reference

GNU sequential classes for public use.

### 4.9.1 Detailed Description

GNU sequential classes for public use.

## 4.10 abi Namespace Reference

The cross-vendor C++ Application Binary Interface. A namespace alias to `__cxxabiv1`, but user programs should use the alias `'abi'`.

### 4.10.1 Detailed Description

The cross-vendor C++ Application Binary Interface. A namespace alias to `__cxxabiv1`, but user programs should use the alias `'abi'`. A brief overview of an ABI is given in the `libstdc++` FAQ, question 5.8 (you may have a copy of the FAQ locally, or you can view the online version at <http://gcc.gnu.org/onlinedocs/libstdc++/faq/index.html#5-8>).

GCC subscribes to a cross-vendor ABI for C++, sometimes called the IA64 ABI because it happens to be the native ABI for that platform. It is summarized at <http://www.codesourcery.com/cxx-abi/> along with the current specification.

For users of GCC greater than or equal to 3.x, entry points are available in `<cxxabi.h>`, which notes, *'It is not normally necessary for user programs to include this header, or use the entry points directly. However, this header is available should that be needed.'*

## 4.11 std Namespace Reference

ISO C++ entities toplevel namespace is `std`.

### Namespaces

- namespace `__debug`
- namespace `__detail`
- namespace `__parallel`
- namespace `__profile`
- namespace `chrono`
- namespace `decimal`
- namespace `placeholders`
- namespace `regex_constants`
- namespace `rel_ops`

- namespace [this\\_thread](#)
- namespace [tr1](#)

## Classes

- struct [\\_\\_atomic\\_flag\\_base](#)  
*Base type for `atomic_flag`.*
- class [\\_\\_basic\\_future](#)  
*Common implementation for `future` and `shared_future`.*
- class [\\_\\_codecvt\\_abstract\\_base](#)  
*Common base for `codecvt` functions.*
- class [\\_\\_ctype\\_abstract\\_base](#)  
*Common base for `ctype` facet.*
- struct [\\_\\_declval\\_protector](#)  
*`declval`*
- struct [\\_\\_future\\_base](#)  
*Base class and enclosing scope.*
- class [\\_\\_has\\_iterator\\_category\\_helper](#)  
*Traits class for iterators.*
- struct [\\_\\_is\\_location\\_invariant](#)
- struct [\\_\\_is\\_member\\_pointer\\_helper](#)  
*`is_member_pointer`*
- struct [\\_\\_numeric\\_limits\\_base](#)  
*Part of `std::numeric_limits`.*
- struct [\\_Base\\_bitset](#)
- struct [\\_Base\\_bitset< 0 >](#)
- struct [\\_Base\\_bitset< 1 >](#)
- struct [\\_Build\\_index\\_tuple](#)  
*Builds an `_Index_tuple<0, 1, 2, ..., _Num-1>`.*
- class [\\_Deque\\_base](#)
- struct [\\_Deque\\_iterator](#)  
*A `deque::iterator`.*

- struct [\\_Derives\\_from\\_binary\\_function](#)  
*Determines if the type `_Tp` derives from `binary_function`.*
- struct [\\_Derives\\_from\\_unary\\_function](#)  
*Determines if the type `_Tp` derives from `unary_function`.*
- class [\\_Function\\_base](#)  
*Base class of all polymorphic function object wrappers.*
- struct [\\_Function\\_to\\_function\\_pointer](#)  
*Turns a function type into a function pointer type.*
- struct [\\_Fwd\\_list\\_base](#)  
*Base class for `forward_list`.*
- struct [\\_Fwd\\_list\\_const\\_iterator](#)  
*A `forward_list::const_iterator`.*
- struct [\\_Fwd\\_list\\_iterator](#)  
*A `forward_list::iterator`.*
- struct [\\_Fwd\\_list\\_node](#)  
*A helper node class for `forward_list`. This is just a linked list with a data value in each node. There is a sorting utility method.*
- struct [\\_Fwd\\_list\\_node\\_base](#)  
*A helper basic node class for `forward_list`. This is just a linked list with nothing inside it. There are purely list shuffling utility methods here.*
- struct [\\_Index\\_tuple](#)
- class [\\_List\\_base](#)  
*See `bits/stl_deque.h`'s `_Deque_base` for an explanation.*
- struct [\\_List\\_const\\_iterator](#)  
*A `list::const_iterator`.*
- struct [\\_List\\_iterator](#)  
*A `list::iterator`.*
- struct [\\_List\\_node](#)  
*An actual node in the list.*

- struct [\\_Maybe\\_get\\_result\\_type](#)  
*If we have found a result\_type, extract it.*
- struct [\\_Maybe\\_unary\\_or\\_binary\\_function](#)
- struct [\\_Maybe\\_unary\\_or\\_binary\\_function< \\_Res, \\_T1 >](#)  
*Derives from [unary\\_function](#), as appropriate.*
- struct [\\_Maybe\\_unary\\_or\\_binary\\_function< \\_Res, \\_T1, \\_T2 >](#)  
*Derives from [binary\\_function](#), as appropriate.*
- struct [\\_Maybe\\_wrap\\_member\\_pointer](#)
- struct [\\_Maybe\\_wrap\\_member\\_pointer< \\_Tp \\_Class::\\* >](#)
- class [\\_Mem\\_fn< \\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\) const >](#)  
*Implementation of `mem_fn` for const member function pointers.*
- class [\\_Mem\\_fn< \\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\) const volatile >](#)  
*Implementation of `mem_fn` for const volatile member function pointers.*
- class [\\_Mem\\_fn< \\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\) volatile >](#)  
*Implementation of `mem_fn` for volatile member function pointers.*
- class [\\_Mem\\_fn< \\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\) >](#)  
*Implementation of `mem_fn` for member function pointers.*
- class [\\_Mu< \\_Arg, false, false >](#)
- class [\\_Mu< \\_Arg, false, true >](#)
- class [\\_Mu< \\_Arg, true, false >](#)
- class [\\_Mu< reference\\_wrapper< \\_Tp >, false, false >](#)
- struct [\\_Placeholder](#)  
*The type of placeholder objects defined by `libstdc++`.*
- struct [\\_Reference\\_wrapper\\_base](#)
- struct [\\_Safe\\_tuple\\_element](#)
- struct [\\_Safe\\_tuple\\_element\\_impl](#)
- struct [\\_Safe\\_tuple\\_element\\_impl< \\_\\_i, \\_Tuple, false >](#)
- class [\\_Temporary\\_buffer](#)
- struct [\\_Tuple\\_impl< \\_Idx >](#)
- struct [\\_Tuple\\_impl< \\_Idx, \\_Head, \\_Tail...>](#)
- struct [\\_Vector\\_base](#)  
*See `bits/stl_deque.h`'s `_Deque_base` for an explanation.*
- struct [\\_Weak\\_result\\_type](#)



- struct `_Weak_result_type_impl`
- struct `_Weak_result_type_impl< _Res(&)(_ArgTypes...)>`  
*Retrieve the result type for a function reference.*
- struct `_Weak_result_type_impl< _Res(*)(_ArgTypes...)>`  
*Retrieve the result type for a function pointer.*
- struct `_Weak_result_type_impl< _Res(_ArgTypes...)>`  
*Retrieve the result type for a function type.*
- struct `_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) const >`  
*Retrieve result type for a const member function pointer.*
- struct `_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) const volatile >`  
*Retrieve result type for a const volatile member function pointer.*
- struct `_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) volatile >`  
*Retrieve result type for a volatile member function pointer.*
- struct `_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...)>`  
*Retrieve result type for a member function pointer.*
- struct `add_const`  
*add\_const*
- struct `add_cv`  
*add\_cv*
- struct `add_lvalue_reference`  
*add\_lvalue\_reference*
- struct `add_pointer`  
*add\_pointer*
- struct `add_rvalue_reference`  
*add\_rvalue\_reference*
- struct `add_volatile`  
*add\_volatile*
- struct `adopt_lock_t`

*Assume the calling thread has already obtained mutex ownership /// and manage it.*

- struct `aligned_storage`

*Alignment type.*

- struct `alignment_of`  
`alignment_of`

- class `allocator`

*The standard allocator, as per [20.4].*

*Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt04ch11.html>.*

- class `allocator< void >`

*allocator<void> specialization.*

- struct `allocator_arg_t`

*[allocator.tag]*

- struct `array`

*A standard container for storing a fixed size sequence of elements.*

- struct `atomic`

*atomic /// 29.4.3, Generic atomic type, primary class template.*

- struct `atomic< _Tp * >`

*Partial specialization for pointer types.*

- struct `atomic< bool >`

*Explicit specialization for bool.*

- struct `atomic< char >`

*Explicit specialization for char.*

- struct `atomic< char16_t >`

*Explicit specialization for char16\_t.*

- struct `atomic< char32_t >`

*Explicit specialization for char32\_t.*

- struct `atomic< int >`

*Explicit specialization for int.*

- struct `atomic< long >`

*Explicit specialization for long.*

- struct `atomic< long long >`

*Explicit specialization for long long.*

- struct `atomic< short >`

*Explicit specialization for short.*

- struct `atomic< signed char >`

*Explicit specialization for signed char.*

- struct `atomic< unsigned char >`

*Explicit specialization for unsigned char.*

- struct `atomic< unsigned int >`

*Explicit specialization for unsigned int.*

- struct `atomic< unsigned long >`

*Explicit specialization for unsigned long.*

- struct `atomic< unsigned long long >`

*Explicit specialization for unsigned long long.*

- struct `atomic< unsigned short >`

*Explicit specialization for unsigned short.*

- struct `atomic< wchar_t >`

*Explicit specialization for wchar\_t.*

- struct `atomic_bool`

*atomic\_bool*

- class `auto_ptr`

*A simple smart pointer providing strict ownership semantics.*

- struct `auto_ptr_ref`

- class `back_insert_iterator`

*Turns assignment into insertion.*

- class `bad_alloc`

*Exception possibly thrown by new.*

*bad\_alloc (or classes derived from it) is used to report allocation errors from the throwing forms of new.*

- class `bad_cast`  
*Thrown during incorrect typecasting.  
If you attempt an invalid `dynamic_cast` expression, an instance of this class (or something derived from this class) is thrown.*
- class `bad_exception`
- class `bad_function_call`  
*Exception class thrown when class template function's `operator()` is called with an empty target.*
- class `bad_typeid`  
*Thrown when a `NULL` pointer in a `typeid` expression is used.*
- class `bad_weak_ptr`  
*Exception possibly thrown by `shared_ptr`.*
- class `basic_filebuf`  
*The actual work of input and output (for files).  
This class associates both its input and output sequence with an external disk file, and maintains a joint file position for both sequences. Many of its semantics are described in terms of similar behavior in the Standard C Library's `FILE` streams.*
- class `basic_fstream`  
*Controlling input and output for files.  
This class supports reading from and writing to named files, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.*
- class `basic_ifstream`  
*Controlling input for files.  
This class supports reading from named files, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.*
- class `basic_ios`  
*Virtual base class for all stream classes.  
Most of the member functions called dispatched on stream objects (e.g., `std::cout.foo(bar);`) are consolidated in this class.*
- class `basic_istream`  
*Merging `istream` and `ostream` capabilities.  
This class multiply inherits from the input and output stream classes simply to provide a single interface.*

- class `basic_istream`  
*Controlling input.*  
*This is the base class for all input streams. It provides text formatting of all builtin types, and communicates with any class derived from `basic_streambuf` to do the actual input.*
- class `basic_istreamstream`  
*Controlling input for `std::string`.*  
*This class supports reading from objects of type `std::basic_string`, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.*
- class `basic_ofstream`  
*Controlling output for files.*  
*This class supports reading from named files, using the inherited functions from `std::basic_ostream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.*
- class `basic_ostream`  
*Controlling output.*  
*This is the base class for all output streams. It provides text formatting of all builtin types, and communicates with any class derived from `basic_streambuf` to do the actual output.*
- class `basic_ostreamstream`  
*Controlling output for `std::string`.*  
*This class supports writing to objects of type `std::basic_string`, using the inherited functions from `std::basic_ostream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.*
- class `basic_regex`
- class `basic_streambuf`  
*The actual work of input and output (interface).*  
*This is a base class. Derived stream buffers each control a pair of character sequences: one for input, and one for output.*
- class `basic_string`  
*Managing sequences of characters and character-like objects.*
- class `basic_stringbuf`  
*The actual work of input and output (for `std::string`).*  
*This class associates either or both of its input and output sequences with a sequence of characters, which can be initialized from, or made available as, a `std::basic_string`. (Paraphrased from [27.7.1]/1.).*

- class [basic\\_stringstream](#)  
*Controlling input and output for `std::string`.  
This class supports reading from and writing to objects of type `std::basic_string`, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.*
- class [bernoulli\\_distribution](#)  
*A Bernoulli random number distribution.*
- struct [bidirectional\\_iterator\\_tag](#)  
*Bidirectional iterators support a superset of forward iterator /// operations.*
- struct [binary\\_function](#)
- class [binary\\_negate](#)  
*One of the *negation functors*.*
- class [binder1st](#)  
*One of the *binder functors*.*
- class [binder2nd](#)  
*One of the *binder functors*.*
- class [binomial\\_distribution](#)  
*A discrete binomial random number distribution.*
- class [bitset](#)  
*The `bitset` class represents a fixed-size sequence of bits.*
- class [cauchy\\_distribution](#)  
*A *cauchy\_distribution* random number distribution.*
- struct [char\\_traits](#)  
*Basis for explicit traits specializations.*
- struct [char\\_traits< \\_\\_gnu\\_cxx::character< V, I, S > >](#)  
*`char_traits<__gnu_cxx::character>` specialization.*
- struct [char\\_traits< char >](#)  
*21.1.3.1 *char\_traits* specializations*
- struct [char\\_traits< wchar\\_t >](#)  
*21.1.3.2 *char\_traits* specializations*

- class [chi\\_squared\\_distribution](#)  
*A [chi\\_squared\\_distribution](#) random number distribution.*
- class [codecvt](#)  
*Primary class template [codecvt](#).  
NB: Generic, mostly useless implementation.*
- class [codecvt<\\_InternT, \\_ExternT, encoding\\_state >](#)  
*[codecvt<InternT, \\_ExternT, encoding\\_state>](#) specialization.*
- class [codecvt<char, char, mbstate\\_t >](#)  
*class [codecvt<char, char, mbstate\\_t>](#) specialization.*
- class [codecvt<wchar\\_t, char, mbstate\\_t >](#)  
*class [codecvt<wchar\\_t, char, mbstate\\_t>](#) specialization.*
- class [codecvt\\_base](#)  
*Empty base class for [codecvt](#) facet [22.2.1.5].*
- class [codecvt\\_byname](#)  
*class [codecvt\\_byname](#) [22.2.1.6].*
- class [collate](#)  
*Facet for localized string comparison.*
- class [collate\\_byname](#)  
*class [collate\\_byname](#) [22.2.4.2].*
- struct [complex](#)
- class [condition\\_variable](#)  
*[condition\\_variable](#)*
- class [condition\\_variable\\_any](#)  
*[condition\\_variable\\_any](#)*
- struct [conditional](#)  
*[conditional](#)*
- class [const\\_mem\\_fun1\\_ref\\_t](#)  
*One of the [adaptors](#) for member /// pointers.*
- class [const\\_mem\\_fun1\\_t](#)

One of the *adaptors for member /// pointers*.

- class `const_mem_fun_ref_t`

One of the *adaptors for member /// pointers*.

- class `const_mem_fun_t`

One of the *adaptors for member /// pointers*.

- class `ctype`

Primary class template `ctype` facet.

This template class defines classification and conversion functions for character sets.

It wraps `cctype` functionality. `Ctype` gets used by streams for many I/O operations.

- class `ctype< char >`

The `ctype<char>` specialization.

This class defines classification and conversion functions for the `char` type. It gets used by `char` streams for many I/O operations. The `char` specialization provides a number of optimizations as well.

- class `ctype< wchar_t >`

The `ctype<wchar_t>` specialization.

This class defines classification and conversion functions for the `wchar_t` type. It gets used by `wchar_t` streams for many I/O operations. The `wchar_t` specialization provides a number of optimizations as well.

- struct `ctype_base`

Base class for `ctype`.

- class `ctype_byname`

class `ctype_byname` [22.2.1.2].

- class `ctype_byname< char >`

22.2.1.4 Class `ctype_byname` specializations.

- class `decay`

*decay*

- struct `default_delete`

Primary template, *default\_delete*.

- struct `default_delete< _Tp[ ]>`

Specialization, *default\_delete*.

- struct `defer_lock_t`



*Do not acquire ownership of the mutex.*

- class `deque`  
*A standard container using fixed-size memory allocation and constant-time manipulation of elements at either end.*
- class `discard_block_engine`
- class `discrete_distribution`  
*A `discrete_distribution` random number distribution.*
- struct `divides`  
*One of the `math functors`.*
- class `domain_error`
- struct `enable_if`  
*`enable_if`*
- class `enable_shared_from_this`  
*Base class allowing use of member function `shared_from_this`.*
- struct `equal_to`  
*One of the `comparison functors`.*
- class `error_category`  
*`error_category`*
- struct `error_code`  
*`error_code`*
- struct `error_condition`  
*`error_condition`*
- class `exception`  
*Base class for all library exceptions.*
- class `exponential_distribution`  
*An exponential continuous distribution for random numbers.*
- struct `extent`  
*`extent`*
- class `extreme_value_distribution`  
*A `extreme_value_distribution` random number distribution.*

- class [fisher\\_f\\_distribution](#)  
*A [fisher\\_f\\_distribution](#) random number distribution.*
- struct [forward\\_iterator\\_tag](#)  
*Forward iterators support a superset of input iterator operations.*
- class [forward\\_list](#)  
*A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.*
- class [fpos](#)  
*Class representing stream positions.*
- class [front\\_insert\\_iterator](#)  
*Turns assignment into insertion.*
- class [function< \\_Res\(\\_ArgTypes...\)>](#)  
*Primary class template for `std::function`.  
Polymorphic function wrapper.*
- class [future](#)  
*Primary template for future.*
- class [future< \\_Res & >](#)  
*Partial specialization for `future<R&>`*
- class [future< void >](#)  
*Explicit specialization for `future<void>`*
- class [future\\_error](#)  
*Exception type thrown by futures.*
- class [gamma\\_distribution](#)  
*A gamma continuous distribution for random numbers.*
- class [geometric\\_distribution](#)  
*A discrete geometric random number distribution.*
- struct [greater](#)  
*One of the [comparison functors](#).*
- struct [greater\\_equal](#)

One of the *comparison functors*.

- class [gslice](#)  
*Class defining multi-dimensional subset of an array.*
- class [gslice\\_array](#)  
*Reference to multi-dimensional subset of an array.*
- struct [has\\_nothrow\\_copy\\_assign](#)  
*has\_nothrow\_copy\_assign*
- struct [has\\_nothrow\\_copy\\_constructor](#)  
*has\_nothrow\_copy\_constructor*
- struct [has\\_nothrow\\_default\\_constructor](#)  
*has\_nothrow\_default\_constructor*
- struct [has\\_trivial\\_copy\\_assign](#)  
*has\_trivial\_copy\_assign*
- struct [has\\_trivial\\_copy\\_constructor](#)  
*has\_trivial\_copy\_constructor*
- struct [has\\_trivial\\_default\\_constructor](#)  
*has\_trivial\_default\_constructor*
- struct [has\\_trivial\\_destructor](#)  
*has\_trivial\_destructor*
- struct [has\\_virtual\\_destructor](#)  
*has\_virtual\_destructor*
- struct [hash](#)  
*Primary class template hash.*
- struct [hash< \\_\\_debug::bitset< \\_Nb > >](#)  
*std::hash specialization for bitset.*
- struct [hash< \\_\\_debug::vector< bool, \\_Alloc > >](#)  
*std::hash specialization for vector<bool>.*
- struct [hash< \\_\\_gnu\\_cxx::throw\\_value\\_limit >](#)

Explicit specialization of `std::hash` for `__gnu_cxx::throw_value_limit`.

- struct `hash< __gnu_cxx::throw_value_random >`  
*Explicit specialization of `std::hash` for `__gnu_cxx::throw_value_limit`.*
- struct `hash< __profile::bitset< _Nb > >`  
*`std::hash` specialization for `bitset`.*
- struct `hash< __profile::vector< bool, _Alloc > >`  
*`std::hash` specialization for `vector<bool>`.*
- struct `hash< __shared_ptr< _Tp, _Lp > >`  
*`std::hash` specialization for `__shared_ptr`.*
- struct `hash< _Tp * >`  
*Partial specializations for pointer types.*
- struct `hash< error_code >`  
*`std::hash` specialization for `error_code`.*
- struct `hash< shared_ptr< _Tp > >`  
*`std::hash` specialization for `shared_ptr`.*
- struct `hash< string >`  
*`std::hash` specialization for `string`.*
- struct `hash< thread::id >`  
*`std::hash` specialization for `thread::id`.*
- struct `hash< type_index >`  
*`std::hash` specialization for `type_index`.*
- struct `hash< u16string >`  
*`std::hash` specialization for `u16string`.*
- struct `hash< u32string >`  
*`std::hash` specialization for `u32string`.*
- struct `hash< unique_ptr< _Tp, _Dp > >`  
*`std::hash` specialization for `unique_ptr`.*
- struct `hash< wstring >`

*std::hash specialization for wstring.*

- struct `hash<::bitset< _Nb > >`

*std::hash specialization for bitset.*

- struct `hash<::vector< bool, _Alloc > >`

*std::hash specialization for vector<bool>.*

- class `independent_bits_engine`

- class `indirect_array`

*Reference to arbitrary subset of an array.*

- class `initializer_list`

*initializer\_list*

- struct `input_iterator_tag`

*Marking input iterators.*

- class `insert_iterator`

*Turns assignment into insertion.*

- struct `integral_constant`

*integral\_constant*

- class `invalid_argument`

- class `ios_base`

*The base of the I/O class hierarchy.*

*This class defines everything that can be defined about I/O that does not depend on the type of characters being input or output. Most people will only see `ios_base` when they need to specify the full name of the various I/O flags (e.g., the openmodes).*

- struct `is_abstract`

*is\_abstract*

- struct `is_arithmetic`

*is\_arithmetic*

- struct `is_array`

*is\_array*

- struct `is_base_of`

*is\_base\_of*

- struct [is\\_bind\\_expression](#)  
*Determines if the given type `_Tp` is a function object should be treated as a subexpression when evaluating calls to function objects returned by [bind\(\)](#). [TR1 3.6.1].*
- struct [is\\_bind\\_expression< \\_Bind< \\_Signature > >](#)  
*Class template `_Bind` is always a bind expression.*
- struct [is\\_bind\\_expression< \\_Bind\\_result< \\_Result, \\_Signature > >](#)  
*Class template `_Bind` is always a bind expression.*
- struct [is\\_class](#)  
*is\_class*
- struct [is\\_compound](#)  
*is\_compound*
- struct [is\\_const](#)  
*is\_const*
- struct [is\\_constructible](#)  
*is\_constructible*
- struct [is\\_convertible](#)  
*is\_convertible*
- struct [is\\_empty](#)  
*is\_empty*
- struct [is\\_enum](#)  
*is\_enum*
- struct [is\\_error\\_code\\_enum](#)  
*is\_error\_code\_enum*
- struct [is\\_error\\_code\\_enum< future\\_errc >](#)  
*Specialization.*
- struct [is\\_error\\_condition\\_enum](#)  
*is\_error\_condition\_enum*
- struct [is\\_explicitly\\_convertible](#)  
*is\_explicitly\_convertible*

- struct [is\\_floating\\_point](#)  
*is\_floating\_point*
- struct [is\\_function](#)  
*is\_function*
- struct [is\\_fundamental](#)  
*is\_fundamental*
- struct [is\\_integral](#)  
*is\_integral*
- struct [is\\_literal\\_type](#)  
*is\_literal\_type*
- struct [is\\_lvalue\\_reference](#)  
*is\_lvalue\_reference*
- struct [is\\_member\\_function\\_pointer](#)  
*is\_member\_function\_pointer*
- struct [is\\_member\\_object\\_pointer](#)  
*is\_member\_object\_pointer*
- struct [is\\_nothrow\\_constructible](#)  
*is\_nothrow\_constructible*
- struct [is\\_object](#)  
*is\_object*
- struct [is\\_placeholder](#)  
*Determines if the given type `_Tp` is a placeholder in a [bind\(\)](#) expression and, if so, which placeholder it is. [TR1 3.6.2].*
- struct [is\\_placeholder< \\_Placeholder< \\_Num > >](#)
- struct [is\\_pod](#)  
*is\_pod*
- struct [is\\_pointer](#)  
*is\_pointer*
- struct [is\\_polymorphic](#)

*is\_polymorphic*

- struct `is_reference`

*is\_reference*

- struct `is_rvalue_reference`

*is\_rvalue\_reference*

- struct `is_same`

*is\_same*

- struct `is_scalar`

*is\_scalar*

- struct `is_signed`

*is\_signed*

- struct `is_standard_layout`

*is\_standard\_layout*

- struct `is_trivial`

*is\_trivial*

- struct `is_union`

*is\_union*

- struct `is_unsigned`

*is\_unsigned*

- struct `is_void`

*is\_void*

- struct `is_volatile`

*is\_volatile*

- class `istream_iterator`

*Provides input iterator semantics for streams.*

- class `istreambuf_iterator`

*Provides input iterator semantics for streambufs.*

- struct `iterator`



*Common iterator class.*

- struct `iterator_traits< _Tp * >`

*Partial specialization for pointer types.*

- struct `iterator_traits< const _Tp * >`

*Partial specialization for const pointer types.*

- class `length_error`

- struct `less`

*One of the [comparison functors](#).*

- struct `less_equal`

*One of the [comparison functors](#).*

- class `linear_congruential_engine`

*A model of a linear congruential random number generator.*

- class `list`

*A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.*

- class `locale`

*Container class for localization functionality.*

*The locale class is first a class wrapper for C library locales. It is also an extensible container for user-defined localization. A locale is a collection of facets that implement various localization features such as money, time, and number printing.*

- class `lock_guard`

*Scoped lock idiom.*

- class `logic_error`

*One of two subclasses of exception.*

- struct `logical_and`

*One of the [Boolean operations functors](#).*

- struct `logical_not`

*One of the [Boolean operations functors](#).*

- struct `logical_or`

*One of the [Boolean operations functors](#).*

- class [lognormal\\_distribution](#)  
*A [lognormal\\_distribution](#) random number distribution.*
- struct [make\\_signed](#)  
*[make\\_signed](#)*
- struct [make\\_unsigned](#)  
*[make\\_unsigned](#)*
- class [map](#)  
*A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.*
- class [mask\\_array](#)  
*Reference to selected subset of an array.*
- class [match\\_results](#)  
*The results of a match or search operation.*
- class [mem\\_fun1\\_ref\\_t](#)  
*One of the [adaptors](#) for member /// pointers.*
- class [mem\\_fun1\\_t](#)  
*One of the [adaptors](#) for member /// pointers.*
- class [mem\\_fun\\_ref\\_t](#)  
*One of the [adaptors](#) for member /// pointers.*
- class [mem\\_fun\\_t](#)  
*One of the [adaptors](#) for member /// pointers.*
- class [messages](#)  
*Primary class template messages.  
This facet encapsulates the code to retrieve messages from message catalogs. The only thing defined by the standard for this facet is the interface. All underlying functionality is implementation-defined.*
- struct [messages\\_base](#)  
*Messages facet base class providing catalog typedef.*
- class [messages\\_byname](#)  
*class [messages\\_byname](#) [22.2.7.2].*

- struct [minus](#)  
*One of the [math functors](#).*
- struct [modulus](#)  
*One of the [math functors](#).*
- class [money\\_base](#)  
*Money format ordering data.  
This class contains an ordered array of 4 fields to represent the pattern for formatting a money amount. Each field may contain one entry from the part enum. symbol, sign, and value must be present and the remaining field must contain either none or space.*
- class [money\\_get](#)  
*Primary class template [money\\_get](#).  
This facet encapsulates the code to parse and return a monetary amount from a string.*
- class [money\\_put](#)  
*Primary class template [money\\_put](#).  
This facet encapsulates the code to format and output a monetary amount.*
- class [moneypunct](#)  
*Primary class template [moneypunct](#).  
This facet encapsulates the punctuation, grouping and other formatting features of money amount string representations.*
- class [moneypunct\\_byname](#)  
*class [moneypunct\\_byname](#) [22.2.6.4].*
- class [move\\_iterator](#)
- class [multimap](#)  
*A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.*
- struct [multiplies](#)  
*One of the [math functors](#).*
- class [multiset](#)  
*A standard container made up of elements, which can be retrieved in logarithmic time.*
- class [mutex](#)  
*mutex*
- struct [negate](#)  
*One of the [math functors](#).*

- class [negative\\_binomial\\_distribution](#)  
*A [negative\\_binomial\\_distribution](#) random number distribution.*
- class [nested\\_exception](#)  
*Exception class with `exception_ptr` data member.*
- class [normal\\_distribution](#)  
*A normal continuous distribution for random numbers.*
- struct [not\\_equal\\_to](#)  
*One of the [comparison functors](#).*
- class [num\\_get](#)  
*Primary class template [num\\_get](#).  
This facet encapsulates the code to parse and return a number from a string. It is used by the istream numeric extraction operators.*
- class [num\\_put](#)  
*Primary class template [num\\_put](#).  
This facet encapsulates the code to convert a number to a string. It is used by the ostream numeric insertion operators.*
- struct [numeric\\_limits](#)  
*Properties of fundamental types.*
- struct [numeric\\_limits< bool >](#)  
*[numeric\\_limits<bool>](#) specialization.*
- struct [numeric\\_limits< char >](#)  
*[numeric\\_limits<char>](#) specialization.*
- struct [numeric\\_limits< char16\\_t >](#)  
*[numeric\\_limits<char16\\_t>](#) specialization.*
- struct [numeric\\_limits< char32\\_t >](#)  
*[numeric\\_limits<char32\\_t>](#) specialization.*
- struct [numeric\\_limits< double >](#)  
*[numeric\\_limits<double>](#) specialization.*
- struct [numeric\\_limits< float >](#)  
*[numeric\\_limits<float>](#) specialization.*

- struct `numeric_limits< int >`  
*`numeric_limits<int>` specialization.*
- struct `numeric_limits< long >`  
*`numeric_limits<long>` specialization.*
- struct `numeric_limits< long double >`  
*`numeric_limits<long double>` specialization.*
- struct `numeric_limits< long long >`  
*`numeric_limits<long long>` specialization.*
- struct `numeric_limits< short >`  
*`numeric_limits<short>` specialization.*
- struct `numeric_limits< signed char >`  
*`numeric_limits<signed char>` specialization.*
- struct `numeric_limits< unsigned char >`  
*`numeric_limits<unsigned char>` specialization.*
- struct `numeric_limits< unsigned int >`  
*`numeric_limits<unsigned int>` specialization.*
- struct `numeric_limits< unsigned long >`  
*`numeric_limits<unsigned long>` specialization.*
- struct `numeric_limits< unsigned long long >`  
*`numeric_limits<unsigned long long>` specialization.*
- struct `numeric_limits< unsigned short >`  
*`numeric_limits<unsigned short>` specialization.*
- struct `numeric_limits< wchar_t >`  
*`numeric_limits<wchar_t>` specialization.*
- class `numpunct`  
*Primary class template `numpunct`.  
This facet stores several pieces of information related to printing and scanning numbers, such as the decimal point character. It takes a template parameter specifying the char type. The `numpunct` facet is used by streams for many I/O operations involving numbers.*

- class `numpunct_byname`  
*class `numpunct_byname` [22.2.3.2].*
- struct `once_flag`  
*`once_flag`*
- class `ostream_iterator`  
*Provides output iterator semantics for streams.*
- class `ostreambuf_iterator`  
*Provides output iterator semantics for streambufs.*
- class `out_of_range`
- struct `output_iterator_tag`  
*Marking output iterators.*
- class `overflow_error`
- struct `owner_less< shared_ptr< _Tp > >`  
*Partial specialization of `owner_less` for `shared_ptr`.*
- struct `owner_less< weak_ptr< _Tp > >`  
*Partial specialization of `owner_less` for `weak_ptr`.*
- class `packaged_task< _Res(_ArgTypes...)>`  
*`packaged_task`*
- struct `pair`  
*Struct holding two objects of arbitrary type.*
- class `piecewise_constant_distribution`  
*A `piecewise_constant_distribution` random number distribution.*
- struct `piecewise_construct_t`  
*`piecewise_construct_t`*
- class `piecewise_linear_distribution`  
*A `piecewise_linear_distribution` random number distribution.*
- struct `plus`  
*One of the `math` functors.*

- class `pointer_to_binary_function`  
*One of the [adaptors for function pointers](#).*
- class `pointer_to_unary_function`  
*One of the [adaptors for function pointers](#).*
- class `poisson_distribution`  
*A discrete Poisson random number distribution.*
- class `priority_queue`  
*A standard container automatically sorting its contents.*
- class `promise`  
*Primary template for promise.*
- class `promise< _Res & >`  
*Partial specialization for `promise<R&>`*
- class `promise< void >`  
*Explicit specialization for `promise<void>`*
- class `queue`  
*A standard container giving FIFO behavior.*
- struct `random_access_iterator_tag`  
*Random-access iterators support a superset of bidirectional /// iterator operations.*
- class `random_device`
- class `range_error`
- struct `rank`  
*rank*
- struct `ratio`  
*Provides compile-time rational arithmetic.*
- struct `ratio_add`  
*ratio\_add*
- struct `ratio_divide`  
*ratio\_divide*
- struct `ratio_equal`

*ratio\_equal*

- struct `ratio_multiply`

*ratio\_multiply*

- struct `ratio_not_equal`

*ratio\_not\_equal*

- struct `ratio_subtract`

*ratio\_subtract*

- class `raw_storage_iterator`

- class `recursive_mutex`

*recursive\_mutex*

- class `recursive_timed_mutex`

*recursive\_timed\_mutex*

- class `reference_wrapper`

*Primary class template for `reference_wrapper`.*

- class `regex_error`

*A regular expression exception class.*

*The regular expression library throws objects of this class on error.*

- class `regex_iterator`

- class `regex_token_iterator`

- struct `regex_traits`

*Describes aspects of a regular expression.*

- struct `remove_all_extents`

*remove\_all\_extents*

- struct `remove_const`

*remove\_const*

- struct `remove_cv`

*remove\_cv*

- struct `remove_extent`

*remove\_extent*

- struct `remove_pointer`



*remove\_pointer*

- struct `remove_reference`

*remove\_reference*

- struct `remove_volatile`

*remove\_volatile*

- class `reverse_iterator`

- class `runtime_error`

*One of two subclasses of exception.*

- class `seed_seq`

*The `seed_seq` class generates sequences of seeds for random number generators.*

- class `set`

*A standard container made up of unique keys, which can be retrieved in logarithmic time.*

- class `shared_future`

*Primary template for `shared_future`.*

- class `shared_future< _Res & >`

*Partial specialization for `shared_future<R&>`*

- class `shared_future< void >`

*Explicit specialization for `shared_future<void>`*

- class `shared_ptr`

*A smart pointer with reference-counted copy semantics.*

- class `shuffle_order_engine`

*Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits `___w`.*

- class `slice`

*Class defining one-dimensional subset of an array.*

- class `slice_array`

*Reference to one-dimensional subset of an array.*

- class `stack`

*A standard container giving FILO behavior.*

- class `student_t_distribution`  
*A `student_t_distribution` random number distribution.*
- class `sub_match`
- class `system_error`  
*Thrown to indicate error code of underlying system.*
- class `thread`  
*thread*
- class `time_base`  
*Time format ordering data.  
This class provides an enum representing different orderings of time: day, month, and year.*
- class `time_get`  
*Primary class template `time_get`.  
This facet encapsulates the code to parse and return a date or time from a string. It is used by the istream numeric extraction operators.*
- class `time_get_byname`  
*class `time_get_byname` [22.2.5.2].*
- class `time_put`  
*Primary class template `time_put`.  
This facet encapsulates the code to format and output dates and times according to formats used by `strftime()`.*
- class `time_put_byname`  
*class `time_put_byname` [22.2.5.4].*
- class `timed_mutex`  
*timed\_mutex*
- struct `try_to_lock_t`  
*Try to acquire ownership of the mutex without blocking.*
- class `tuple`  
*tuple*
- class `tuple<_T1>`  
*tuple (1-element).*

- class `tuple< _T1, _T2 >`  
*tuple (2-element), with construction and assignment from a pair.*
- struct `tuple_element< 0, tuple< _Head, _Tail...> >`
- struct `tuple_element< __i, tuple< _Head, _Tail...> >`
- struct `tuple_size< tuple< _Elements...> >`  
*class tuple\_size*
- struct `type_index`  
*The class `type_index` provides a simple wrapper for `type_info` which can be used as an index type in associative containers (23.6) and in unordered associative containers (23.7).*
- class `type_info`  
*Part of RTTI.*
- struct `unary_function`
- class `unary_negate`  
*One of the *negation functors*.*
- class `underflow_error`
- class `uniform_int_distribution`  
*Uniform discrete distribution for random numbers. A discrete random distribution on the range  $[min, max]$  with equal probability throughout the range.*
- class `uniform_real_distribution`  
*Uniform continuous distribution for random numbers.*
- class `unique_lock`  
*unique\_lock*
- class `unique_ptr`  
*20.7.12.2 `unique_ptr` for single objects.*
- class `unique_ptr< _Tp[ ], _Dp >`  
*20.7.12.3 `unique_ptr` for array objects with a runtime length*
- class `unordered_map`  
*A standard container composed of unique keys (containing at most one of each key value) that associates values of another type with the keys.*
- class `unordered_multimap`

*A standard container composed of equivalent keys (possibly containing multiple of each key value) that associates values of another type with the keys.*

- class [unordered\\_multiset](#)

*A standard container composed of equivalent keys (possibly containing multiple of each key value) in which the elements' keys are the elements themselves.*

- class [unordered\\_set](#)

*A standard container composed of unique keys (containing at most one of each key value) in which the elements' keys are the elements themselves.*

- struct [uses\\_allocator](#)

*[allocator.uses.trait]*

- class [valarray](#)

*Smart array designed to support numeric processing.*

- class [vector](#)

*A standard container which offers fixed time access to individual elements in any order.*

- class [vector< bool, \\_Alloc >](#)

*A specialization of vector for booleans which offers fixed time access to individual elements in any order.*

- class [weak\\_ptr](#)

*A smart pointer with weak semantics.*

- class [weibull\\_distribution](#)

*A [weibull\\_distribution](#) random number distribution.*

## Typedefs

- typedef FILE [\\_\\_c\\_file](#)
- typedef \_\_locale\_t [\\_\\_c\\_locale](#)
- typedef \_\_gthread\_mutex\_t [\\_\\_c\\_lock](#)
- typedef unsigned long [\\_Bit\\_type](#)
- typedef \_\_atomic\_base< char > [atomic\\_char](#)
- typedef \_\_atomic\_base< char16\_t > [atomic\\_char16\\_t](#)
- typedef \_\_atomic\_base< char32\_t > [atomic\\_char32\\_t](#)
- typedef \_\_atomic\_base< int > [atomic\\_int](#)
- typedef \_\_atomic\_base< int\_fast16\_t > [atomic\\_int\\_fast16\\_t](#)

- typedef \_\_atomic\_base< int\_fast32\_t > [atomic\\_int\\_fast32\\_t](#)
- typedef \_\_atomic\_base< int\_fast64\_t > [atomic\\_int\\_fast64\\_t](#)
- typedef \_\_atomic\_base< int\_fast8\_t > [atomic\\_int\\_fast8\\_t](#)
- typedef \_\_atomic\_base< int\_least16\_t > [atomic\\_int\\_least16\\_t](#)
- typedef \_\_atomic\_base< int\_least32\_t > [atomic\\_int\\_least32\\_t](#)
- typedef \_\_atomic\_base< int\_least64\_t > [atomic\\_int\\_least64\\_t](#)
- typedef \_\_atomic\_base< int\_least8\_t > [atomic\\_int\\_least8\\_t](#)
- typedef \_\_atomic\_base< intmax\_t > [atomic\\_intmax\\_t](#)
- typedef \_\_atomic\_base< intptr\_t > [atomic\\_intptr\\_t](#)
- typedef \_\_atomic\_base< long long > [atomic\\_llong](#)
- typedef \_\_atomic\_base< long > [atomic\\_long](#)
- typedef \_\_atomic\_base< ptrdiff\_t > [atomic\\_ptrdiff\\_t](#)
- typedef \_\_atomic\_base< signed char > [atomic\\_schar](#)
- typedef \_\_atomic\_base< short > [atomic\\_short](#)
- typedef \_\_atomic\_base< size\_t > [atomic\\_size\\_t](#)
- typedef \_\_atomic\_base< unsigned char > [atomic\\_uchar](#)
- typedef \_\_atomic\_base< unsigned int > [atomic\\_uint](#)
- typedef \_\_atomic\_base< uint\_fast16\_t > [atomic\\_uint\\_fast16\\_t](#)
- typedef \_\_atomic\_base< uint\_fast32\_t > [atomic\\_uint\\_fast32\\_t](#)
- typedef \_\_atomic\_base< uint\_fast64\_t > [atomic\\_uint\\_fast64\\_t](#)
- typedef \_\_atomic\_base< uint\_fast8\_t > [atomic\\_uint\\_fast8\\_t](#)
- typedef \_\_atomic\_base< uint\_least16\_t > [atomic\\_uint\\_least16\\_t](#)
- typedef \_\_atomic\_base< uint\_least32\_t > [atomic\\_uint\\_least32\\_t](#)
- typedef \_\_atomic\_base< uint\_least64\_t > [atomic\\_uint\\_least64\\_t](#)
- typedef \_\_atomic\_base< uint\_least8\_t > [atomic\\_uint\\_least8\\_t](#)
- typedef \_\_atomic\_base< uintmax\_t > [atomic\\_uintmax\\_t](#)
- typedef \_\_atomic\_base< uintptr\_t > [atomic\\_uintptr\\_t](#)
- typedef \_\_atomic\_base< unsigned long long > [atomic\\_ullong](#)
- typedef \_\_atomic\_base< unsigned long > [atomic\\_ulong](#)
- typedef \_\_atomic\_base< unsigned short > [atomic\\_ushort](#)
- typedef \_\_atomic\_base< wchar\_t > [atomic\\_wchar\\_t](#)
- typedef [match\\_results](#)< const char \* > **cmatch**
- typedef [regex\\_iterator](#)< const char \* > **cregex\_iterator**
- typedef [regex\\_token\\_iterator](#)< const char \* > [cregex\\_token\\_iterator](#)
- typedef [sub\\_match](#)< const char \* > [csub\\_match](#)
- typedef [minstd\\_rand0](#) **default\_random\_engine**
- typedef [integral\\_constant](#)< bool, false > [false\\_type](#)
- typedef [basic\\_filebuf](#)< char > [filebuf](#)
- typedef [basic\\_fstream](#)< char > [fstream](#)
- typedef [basic\\_ifstream](#)< char > [ifstream](#)
- typedef [basic\\_ios](#)< char > [ios](#)
- typedef [basic\\_iostream](#)< char > [iostream](#)
- typedef [basic\\_istream](#)< char > [istream](#)

- typedef [basic\\_istream](#)< char > [istream](#)
- typedef [shuffle\\_order\\_engine](#)< [minstd\\_rand0](#), 256 > **knuth\_b**
- typedef enum [std::memory\\_order](#) [memory\\_order](#)
- typedef [linear\\_congruential\\_engine](#)< [uint\\_fast32\\_t](#), 48271UL, 0UL, 2147483647UL > [minstd\\_rand](#)
- typedef [linear\\_congruential\\_engine](#)< [uint\\_fast32\\_t](#), 16807UL, 0UL, 2147483647UL > [minstd\\_rand0](#)
- typedef [mersenne\\_twister\\_engine](#)< [uint\\_fast32\\_t](#), 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL > [mt19937](#)
- typedef [mersenne\\_twister\\_engine](#)< [uint\\_fast64\\_t](#), 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000ULL, 37, 0xfff7eee000000000ULL, 43, 6364136223846793005ULL > [mt19937\\_64](#)
- typedef void(\* [new\\_handler](#) )()
- typedef [basic\\_ofstream](#)< char > [ofstream](#)
- typedef [basic\\_ostream](#)< char > [ostream](#)
- typedef [basic\\_ostringstream](#)< char > [ostringstream](#)
- typedef \_\_PTRDIFF\_TYPE\_\_ [ptrdiff\\_t](#)
- typedef [discard\\_block\\_engine](#)< [ranlux24\\_base](#), 223, 23 > **ranlux24**
- typedef [subtract\\_with\\_carry\\_engine](#)< [uint\\_fast32\\_t](#), 24, 10, 24 > **ranlux24\_base**
- typedef [discard\\_block\\_engine](#)< [ranlux48\\_base](#), 389, 11 > **ranlux48**
- typedef [subtract\\_with\\_carry\\_engine](#)< [uint\\_fast64\\_t](#), 48, 5, 12 > **ranlux48\_base**
  
- typedef [basic\\_regex](#)< char > [regex](#)
- typedef \_\_SIZE\_TYPE\_\_ [size\\_t](#)
- typedef [match\\_results](#)< [string::const\\_iterator](#) > **smatch**
- typedef [regex\\_iterator](#)< [string::const\\_iterator](#) > **sregex\_iterator**
- typedef [regex\\_token\\_iterator](#)< [string::const\\_iterator](#) > [sregex\\_token\\_iterator](#)
- typedef [sub\\_match](#)< [string::const\\_iterator](#) > [ssub\\_match](#)
- typedef [basic\\_streambuf](#)< char > [streambuf](#)
- typedef long long [streamoff](#)
- typedef [fpos](#)< [mbstate\\_t](#) > [streampos](#)
- typedef [ptrdiff\\_t](#) [streamsize](#)
- typedef [basic\\_string](#)< char > **string**
- typedef [basic\\_stringbuf](#)< char > [stringbuf](#)
- typedef [basic\\_stringstream](#)< char > [stringstream](#)
- typedef void(\* [terminate\\_handler](#) )()
- typedef [integral\\_constant](#)< bool, true > [true\\_type](#)
- typedef [fpos](#)< [mbstate\\_t](#) > [u16streampos](#)
- typedef [basic\\_string](#)< [char16\\_t](#) > **u16string**
- typedef [fpos](#)< [mbstate\\_t](#) > [u32streampos](#)

- typedef [basic\\_string](#)< char32\_t > [u32string](#)
- typedef void(\* [unexpected\\_handler](#) )()
- typedef [match\\_results](#)< const wchar\_t \* > [wcmatch](#)
- typedef [regex\\_iterator](#)< const wchar\_t \* > [wcregex\\_iterator](#)
- typedef [regex\\_token\\_iterator](#)< const wchar\_t \* > [wcregex\\_token\\_iterator](#)
- typedef [sub\\_match](#)< const wchar\_t \* > [wcsub\\_match](#)
- typedef [basic\\_filebuf](#)< wchar\_t > [wfilebuf](#)
- typedef [basic\\_fstream](#)< wchar\_t > [wfstream](#)
- typedef [basic\\_ifstream](#)< wchar\_t > [wifstream](#)
- typedef [basic\\_ios](#)< wchar\_t > [wios](#)
- typedef [basic\\_iostream](#)< wchar\_t > [wiostream](#)
- typedef [basic\\_istream](#)< wchar\_t > [wistream](#)
- typedef [basic\\_istreamstream](#)< wchar\_t > [wistreamstream](#)
- typedef [basic\\_ofstream](#)< wchar\_t > [wofstream](#)
- typedef [basic\\_ostream](#)< wchar\_t > [wostream](#)
- typedef [basic\\_ostreamstream](#)< wchar\_t > [wostreamstream](#)
- typedef [basic\\_regex](#)< wchar\_t > [wregex](#)
- typedef [match\\_results](#)< wstring::const\_iterator > [wsmatch](#)
- typedef [regex\\_iterator](#)< wstring::const\_iterator > [wsregex\\_iterator](#)
- typedef [regex\\_token\\_iterator](#)< wstring::const\_iterator > [wsregex\\_token\\_iterator](#)
- typedef [sub\\_match](#)< wstring::const\_iterator > [wssub\\_match](#)
- typedef [basic\\_streambuf](#)< wchar\_t > [wstreambuf](#)
- typedef [fpos](#)< mbstate\_t > [wstreampos](#)
- typedef [basic\\_string](#)< wchar\_t > [wstring](#)
- typedef [basic\\_stringbuf](#)< wchar\_t > [wstringbuf](#)
- typedef [basic\\_stringstream](#)< wchar\_t > [wstringstream](#)

### Enumerations

- enum { [\\_S\\_threshold](#) }
- enum { [\\_S\\_chunk\\_size](#) }
- enum { [\\_S\\_word\\_bit](#) }
- enum [\\_Ios\\_Fmtflags](#) {  
[\\_S\\_boolalpha](#), [\\_S\\_dec](#), [\\_S\\_fixed](#), [\\_S\\_hex](#),  
[\\_S\\_internal](#), [\\_S\\_left](#), [\\_S\\_oct](#), [\\_S\\_right](#),  
[\\_S\\_scientific](#), [\\_S\\_showbase](#), [\\_S\\_showpoint](#), [\\_S\\_showpos](#),  
[\\_S\\_skipws](#), [\\_S\\_unitbuf](#), [\\_S\\_uppercase](#), [\\_S\\_adjustfield](#),  
[\\_S\\_basefield](#), [\\_S\\_floatfield](#), [\\_S\\_ios\\_fmtflags\\_end](#) }

- enum `_Ios_Iostate` {  
`_S_goodbit`, `_S_badbit`, `_S_eofbit`, `_S_failbit`,  
`_S_ios_iostate_end` }
- enum `_Ios_Openmode` {  
`_S_app`, `_S_ate`, `_S_bin`, `_S_in`,  
`_S_out`, `_S_trunc`, `_S_ios_openmode_end` }
- enum `_Ios_Seekdir` { `_S_beg`, `_S_cur`, `_S_end`, `_S_ios_seekdir_end` }
- enum `_Manager_operation` { `__get_type_info`, `__get_functor_ptr`, `__clone_functor`, `__destroy_functor` }
- enum `_Rb_tree_color` { `_S_red`, `_S_black` }
- enum `cv_status` { `no_timeout`, `timeout` }
- enum `errc` {  
`address_family_not_supported`, `address_in_use`, `address_not_available`,  
`already_connected`,  
`argument_list_too_long`, `argument_out_of_domain`, `bad_address`, `bad_file_descriptor`,  
`bad_message`, `broken_pipe`, `connection_aborted`, `connection_already_in_progress`,  
`connection_refused`, `connection_reset`, `cross_device_link`, `destination_address_required`,  
`device_or_resource_busy`, `directory_not_empty`, `executable_format_error`, `file_exists`,  
`file_too_large`, `filename_too_long`, `function_not_supported`, `host_unreachable`,  
`identifier_removed`, `illegal_byte_sequence`, `inappropriate_io_control_operation`, `interrupted`,  
`invalid_argument`, `invalid_seek`, `io_error`, `is_a_directory`,  
`message_size`, `network_down`, `network_reset`, `network_unreachable`,  
`no_buffer_space`, `no_child_process`, `no_link`, `no_lock_available`,  
`no_message_available`, `no_message`, `no_protocol_option`, `no_space_on_device`,  
`no_stream_resources`, `no_such_device_or_address`, `no_such_device`, `no_such_file_or_directory`,  
`no_such_process`, `not_a_directory`, `not_a_socket`, `not_a_stream`,  
`not_connected`, `not_enough_memory`, `not_supported`, `operation_canceled`,  
`operation_in_progress`, `operation_not_permitted`, `operation_not_supported`, `operation_would_block`,  
`owner_dead`, `permission_denied`, `protocol_error`, `protocol_not_supported`,



**read\_only\_file\_system, resource\_deadlock\_would\_occur, resource\_unavailable\_try\_again, result\_out\_of\_range,**  
**state\_not\_recoverable, stream\_timeout, text\_file\_busy, timed\_out,**  
**too\_many\_files\_open\_in\_system, too\_many\_files\_open, too\_many\_links,**  
**too\_many\_symbolic\_link\_levels,**  
**value\_too\_large, wrong\_protocol\_type }**

- enum [float\\_denorm\\_style](#) { [denorm\\_indeterminate](#), [denorm\\_absent](#), [denorm\\_present](#) }
- enum [float\\_round\\_style](#) {  
**round\_indeterminate, round\_toward\_zero, round\_to\_nearest, round\_toward\_infinity,**  
**round\_toward\_neg\_infinity }**
- enum [future\\_errc](#) { **broken\_promise, future\_already\_retrieved, promise\_already\_satisfied, no\_state** }
- enum [future\\_status](#) { **ready, timeout, deferred** }
- enum [launch](#) { **any, async, sync** }
- enum [memory\\_order](#) {  
**memory\_order\_relaxed, memory\_order\_consume, memory\_order\_acquire, memory\_order\_release,**  
**memory\_order\_acq\_rel, memory\_order\_seq\_cst }**

## Functions

- `template<typename _CharT >`  
`_CharT * __add_grouping (_CharT * __s, _CharT __sep, const char * __gbeg,`  
`size_t __gsize, const _CharT * __first, const _CharT * __last)`
- `template<typename _Tp >`  
`_Tp * __addressof (_Tp & __r)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp >`  
`void __adjust_heap (_RandomAccessIterator __first, _Distance __holeIndex,`  
`_Distance __len, _Tp __value)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _Compare >`  
`void __adjust_heap (_RandomAccessIterator __first, _Distance __holeIndex,`  
`_Distance __len, _Tp __value, _Compare __comp)`
- `template<typename _InputIterator, typename _Distance >`  
`void __advance (_InputIterator & __i, _Distance __n, input\_iterator\_tag)`
- `template<typename _BidirectionalIterator, typename _Distance >`  
`void __advance (_BidirectionalIterator & __i, _Distance __n, bidirectional\_iterator\_tag)`
- `template<typename _RandomAccessIterator, typename _Distance >`  
`void __advance (_RandomAccessIterator & __i, _Distance __n, random\_access\_iterator\_tag)`

- `template<typename _Tp, _Lock_policy _Lp, typename _Alloc, typename... _Args>`  
`__shared_ptr< _Tp, _Lp > __allocate_shared (const _Alloc &__a, _Args`  
`&&...__args)`
- `memory_order __calculate_memory_order (memory_order __m)`
- `template<typename _Functor >`  
`_Functor & __callable_functor (_Functor &__f)`
- `template<typename _Member, typename _Class >`  
`_Mem_fn< _Member _Class::* > __callable_functor (_Member _-`  
`Class::*const &__p)`
- `template<typename _Member, typename _Class >`  
`_Mem_fn< _Member _Class::* > __callable_functor (_Member _Class::*&-`  
`_p)`
- `template<typename _Facet >`  
`const _Facet & __check_facet (const _Facet *__f)`
- `template<typename _RandomAccessIterator, typename _Distance >`  
`void __chunk_insertion_sort (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _Distance __chunk_size)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Compare >`  
`void __chunk_insertion_sort (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _Distance __chunk_size, _Compare __comp)`
  
- `template<typename _Tp >`  
`_Tp __complex_abs (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > __complex_acos (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > __complex_acosh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`_Tp __complex_arg (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > __complex_asin (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > __complex_asinh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > __complex_atan (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > __complex_atanh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_cos (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_cosh (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_exp (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_log (const complex< _Tp > &__z)`

- `template<typename _Tp >`  
`complex< _Tp > __complex_pow` (const `complex< _Tp >` &\_\_x, const `complex< _Tp >` &\_\_y)
- `template<typename _Tp >`  
`std::complex< _Tp > __complex_proj` (const `std::complex< _Tp >` &\_\_z)
- `template<typename _Tp >`  
`complex< _Tp > __complex_sin` (const `complex< _Tp >` &\_\_z)
- `template<typename _Tp >`  
`complex< _Tp > __complex_sinh` (const `complex< _Tp >` &\_\_z)
- `template<typename _Tp >`  
`complex< _Tp > __complex_sqrt` (const `complex< _Tp >` &\_\_z)
- `template<typename _Tp >`  
`complex< _Tp > __complex_tan` (const `complex< _Tp >` &\_\_z)
- `template<typename _Tp >`  
`complex< _Tp > __complex_tanh` (const `complex< _Tp >` &\_\_z)
- `int __convert_from_v` (const `__c_locale` &\_\_cloc `__attribute__((__unused__))`,  
`char *`\_\_out, const `int` \_\_size `__attribute__((__unused__))`, const `char *`\_\_fmt,...)
  
- `template<typename _Tp >`  
`void __convert_to_v` (const `char *`, `_Tp` &, `ios_base::iostate` &, const `__c_locale` &) throw ()
- `template<>`  
`void __convert_to_v` (const `char *`, `float` &, `ios_base::iostate` &, const `__c_locale` &) throw ()
- `template<>`  
`void __convert_to_v` (const `char *`, `double` &, `ios_base::iostate` &, const `__c_locale` &) throw ()
- `template<>`  
`void __convert_to_v` (const `char *`, `long double` &, `ios_base::iostate` &, const `__c_locale` &) throw ()
- `template<bool _IsMove, typename _II, typename _OI >`  
`_OI __copy_move_a` (\_II \_\_first, \_II \_\_last, \_OI \_\_result)
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT, char_traits< _CharT > > >::__type __copy_move_a2` (\_CharT \*, \_CharT \*, `ostreambuf_iterator< _CharT, char_traits< _CharT > >`)
  
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT, char_traits< _CharT > > >::__type __copy_move_a2` (const \_CharT \*, const \_CharT \*, `ostreambuf_iterator< _CharT, char_traits< _CharT > >`)
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type __copy_move_a2` (`istreambuf_iterator< _CharT, char_traits< _CharT > >`, `istreambuf_iterator< _CharT, char_traits< _CharT > >`, \_CharT \*)

- `template<bool _IsMove, typename _II, typename _OI >`  
`_OI __copy_move_a2 (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type __copy_move_a2 (_CharT *__first, _CharT *__last, ostreambuf_iterator< _CharT > __result)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type __copy_move_a2 (const _CharT *__first, const _CharT *__last, ostreambuf_iterator< _CharT > __result)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type __copy_move_a2 (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, _CharT *__result)`
- `template<bool _IsMove, typename _BI1, typename _BI2 >`  
`_BI2 __copy_move_backward_a (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<bool _IsMove, typename _BI1, typename _BI2 >`  
`_BI2 __copy_move_backward_a2 (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`  
`_OutputIterator __copy_n (_InputIterator __first, _Size __n, _OutputIterator __result, input_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Size, typename _OutputIterator >`  
`_OutputIterator __copy_n (_RandomAccessIterator __first, _Size __n, _OutputIterator __result, random_access_iterator_tag)`
- `template<typename _CharT, typename _Traits >`  
`streamsize __copy_streambufs (basic_streambuf< _CharT, _Traits > *__sbin, basic_streambuf< _CharT, _Traits > *__sbout)`
- `template<typename _CharT, typename _Traits >`  
`streamsize __copy_streambufs_eof (basic_streambuf< _CharT, _Traits > *, basic_streambuf< _CharT, _Traits > *, bool &)`
- `template<>`  
`streamsize __copy_streambufs_eof (basic_streambuf< char > *__sbin, basic_streambuf< char > *__sbout, bool &__ineof)`
- `template<>`  
`streamsize __copy_streambufs_eof (basic_streambuf< wchar_t > *__sbin, basic_streambuf< wchar_t > *__sbout, bool &__ineof)`
- `size_t __deque_buf_size (size_t __size)`
- `template<typename _InputIterator >`  
`iterator_traits< _InputIterator >::difference_type __distance (_InputIterator __first, _InputIterator __last, input_iterator_tag)`
- `template<typename _RandomAccessIterator >`  
`iterator_traits< _RandomAccessIterator >::difference_type __distance (_RandomAccessIterator __first, _RandomAccessIterator __last, random_access_iterator_tag)`

- `template<typename _Tp1, typename _Tp2 >`  
`void \_\_enable\_shared\_from\_this\_helper (const __shared_count<> &, const`  
`enable\_shared\_from\_this< _Tp1 > *, const _Tp2 *)`
- `template<_Lock_policy _Lp>`  
`void \_\_enable\_shared\_from\_this\_helper (const __shared_count< _Lp > &,...)`
- `template<_Lock_policy _Lp, typename _Tp1, typename _Tp2 >`  
`void \_\_enable\_shared\_from\_this\_helper (const __shared_count< _Lp > &,`  
`const __enable_shared_from_this< _Tp1, _Lp > *, const _Tp2 *)`
- `template<typename _II1, typename _II2 >`  
`bool \_\_equal\_aux (_II1 __first1, _II1 __last1, _II2 __first2)`
- `template<typename _ForwardIterator, typename _Tp >`  
`__gnu_cxx::__enable_if<!\_\_is\_scalar< _Tp >::__value, void >::__type \_\_fill\_a`  
`( _ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< \_\_is\_byte< _Tp >::__value, void >::__type \_\_fill\_a`  
`( _Tp * __first, _Tp * __last, const _Tp &__c)`
- `void \_\_fill\_bvector (_Bit_iterator __first, _Bit_iterator __last, bool __x)`
- `template<typename _OutputIterator, typename _Size, typename _Tp >`  
`__gnu_cxx::__enable_if<!\_\_is\_scalar< _Tp >::__value, _OutputIterator >::__-`  
`type \_\_fill\_n\_a (_OutputIterator __first, _Size __n, const _Tp &__value)`
- `template<typename _Size, typename _Tp >`  
`__gnu_cxx::__enable_if< \_\_is\_byte< _Tp >::__value, _Tp * >::__type \_\_fill-`  
`n\_a (_Tp * __first, _Size __n, const _Tp &__c)`
- `template<typename _RandomAccessIterator >`  
`void \_\_final\_insertion\_sort (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void \_\_final\_insertion\_sort (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _Tp >`  
`_InputIterator \_\_find (_InputIterator __first, _InputIterator __last, const _Tp &__-`  
`__val, input\_iterator\_tag)`
- `template<typename _RandomAccessIterator, typename _Tp >`  
`_RandomAccessIterator \_\_find (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, const _Tp &__val, random\_access\_iterator\_tag)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`_ForwardIterator1 \_\_find\_end (_ForwardIterator1 __first1, _ForwardIterator1`  
`__last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, forward-`  
`iterator\_tag, forward\_iterator\_tag)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate`  
`>`  
`_ForwardIterator1 \_\_find\_end (_ForwardIterator1 __first1, _ForwardIterator1`

- 
- ```

__last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, forward\_
iterator\_tag, forward\_iterator\_tag, _BinaryPredicate __comp)

```
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2 >`
`_BidirectionalIterator1 __find_end (_BidirectionalIterator1 __first1,`
`_BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _`
`BidirectionalIterator2 __last2, bidirectional_iterator_tag, bidirectional_`
`iterator_tag)`
 - `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _`
`BinaryPredicate >`
`_BidirectionalIterator1 __find_end (_BidirectionalIterator1 __first1,`
`_BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _`
`BidirectionalIterator2 __last2, bidirectional_iterator_tag, bidirectional_`
`iterator_tag, _BinaryPredicate __comp)`
 - `template<typename _InputIterator, typename _Predicate >`
`_InputIterator __find_if (_InputIterator __first, _InputIterator __last, _Predicate`
`__pred, input_iterator_tag)`
 - `template<typename _RandomAccessIterator, typename _Predicate >`
`_RandomAccessIterator __find_if (_RandomAccessIterator __first, _`
`RandomAccessIterator __last, _Predicate __pred, random_access_iterator_tag)`
 - `template<typename _InputIterator, typename _Predicate >`
`_InputIterator __find_if_not (_InputIterator __first, _InputIterator __last, _`
`Predicate __pred, input_iterator_tag)`
 - `template<typename _RandomAccessIterator, typename _Predicate >`
`_RandomAccessIterator __find_if_not (_RandomAccessIterator __first, _`
`RandomAccessIterator __last, _Predicate __pred, random_access_iterator_tag)`
 - `template<typename _EuclideanRingElement >`
`_EuclideanRingElement __gcd (_EuclideanRingElement __m, _`
`EuclideanRingElement __n)`
 - `template<std::size_t __i, typename _Head, typename... _Tail>`
`__add_c_ref< _Head >::type __get_helper (const _Tuple_impl< __i, _Head,`
`_Tail...> &__t)`
 - `template<std::size_t __i, typename _Head, typename... _Tail>`
`__add_ref< _Head >::type __get_helper (_Tuple_impl< __i, _Head, _Tail...>`
`&__t)`
 - `template<typename _Ex >`
`const nested_exception * __get_nested_exception (const _Ex &__ex)`
 - `mutex & __get_once_mutex ()`
 - `template<typename _RandomAccessIterator, typename _Compare >`
`void __heap_select (_RandomAccessIterator __first, _RandomAccessIterator _`
`_middle, _RandomAccessIterator __last, _Compare __comp)`
 - `template<typename _RandomAccessIterator >`
`void __heap_select (_RandomAccessIterator __first, _RandomAccessIterator _`
`_middle, _RandomAccessIterator __last)`
-

- `template<typename _Tp >`
`size_t __iconv_adaptor (size_t(*__func)(iconv_t, _Tp, size_t *, char **, size_t *), iconv_t __cd, char **__inbuf, size_t *__inbytes, char **__outbuf, size_t *__outbytes)`
- `template<typename _ForwardIterator, typename _Predicate, typename _Distance >`
`_ForwardIterator __inplace_stable_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, _Distance __len)`
- `template<typename _RandomAccessIterator >`
`void __inplace_stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void __inplace_stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void __insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void __insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _CharT, typename _ValueT >`
`int __int_to_char (_CharT *__bufend, _ValueT __v, const _CharT *__lit, ios_base::fmtflags __flags, bool __dec)`
- `template<typename _RandomAccessIterator, typename _Size >`
`void __introsselect (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Size __depth_limit)`
- `template<typename _RandomAccessIterator, typename _Size, typename _Compare >`
`void __introsselect (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Size __depth_limit, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Size >`
`void __introsort_loop (_RandomAccessIterator __first, _RandomAccessIterator __last, _Size __depth_limit)`
- `template<typename _RandomAccessIterator, typename _Size, typename _Compare >`
`void __introsort_loop (_RandomAccessIterator __first, _RandomAccessIterator __last, _Size __depth_limit, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance >`
`bool __is_heap (_RandomAccessIterator __first, _Distance __n)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`bool __is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare, typename _Distance >`
`bool __is_heap (_RandomAccessIterator __first, _Compare __comp, _Distance __n)`

- `template<typename _RandomAccessIterator >`
`bool __is_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`
`last)`
- `template<typename _RandomAccessIterator, typename _Distance >`
`_Distance __is_heap_until (_RandomAccessIterator __first, _Distance __n)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Compare >`
`_Distance __is_heap_until (_RandomAccessIterator __first, _Distance __n, _-`
`Compare __comp)`
- `template<typename _Iter >`
`iterator_traits< _Iter >::iterator_category __iterator_category (const _Iter &)`
- `template<typename _II1, typename _II2 >`
`bool __lexicographical_compare_aux (_II1 __first1, _II1 __last1, _II2 __first2,`
`_II2 __last2)`
- `template<typename _Size >`
`_Size __lg (_Size __n)`
- `int __lg (int __n)`
- `long __lg (long __n)`
- `long long __lg (long long __n)`
- `template<typename _Tp, _Lock_policy _Lp, typename... _Args>`
`__shared_ptr< _Tp, _Lp > __make_shared (_Args &&...__args)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer >`
`void __merge_adaptive (_BidirectionalIterator __first, _BidirectionalIterator __-`
`middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _-`
`Pointer __buffer, _Distance __buffer_size)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer, typename _-`
`Compare >`
`void __merge_adaptive (_BidirectionalIterator __first, _BidirectionalIterator __-`
`middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _-`
`Pointer __buffer, _Distance __buffer_size, _Compare __comp)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _-`
`BidirectionalIterator3 >`
`_BidirectionalIterator3 __merge_backward (_BidirectionalIterator1 __-`
`first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _-`
`BidirectionalIterator2 __last2, _BidirectionalIterator3 __result)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _-`
`BidirectionalIterator3, typename _Compare >`
`_BidirectionalIterator3 __merge_backward (_BidirectionalIterator1 __-`
`first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _-`
`BidirectionalIterator2 __last2, _BidirectionalIterator3 __result, _Compare`
`__comp)`
- `template<typename _RandomAccessIterator1, typename _RandomAccessIterator2, typename`
`_Distance, typename _Compare >`
`void __merge_sort_loop (_RandomAccessIterator1 __first, _-`
`RandomAccessIterator1 __last, _RandomAccessIterator2 __result, _Distance`
`__step_size, _Compare __comp)`

- `template<typename _RandomAccessIterator1 , typename _RandomAccessIterator2 , typename _Distance >`
`void __merge_sort_loop (_RandomAccessIterator1 __first, _-`
`RandomAccessIterator1 __last, _RandomAccessIterator2 __result, _Distance`
`__step_size)`
- `template<typename _RandomAccessIterator , typename _Pointer >`
`void __merge_sort_with_buffer (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _Pointer __buffer)`
- `template<typename _RandomAccessIterator , typename _Pointer , typename _Compare >`
`void __merge_sort_with_buffer (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _Pointer __buffer, _Compare __comp)`
- `template<typename _BidirectionalIterator , typename _Distance >`
`void __merge_without_buffer (_BidirectionalIterator __first, _-`
`BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance`
`__len1, _Distance __len2)`
- `template<typename _BidirectionalIterator , typename _Distance , typename _Compare >`
`void __merge_without_buffer (_BidirectionalIterator __first, _-`
`BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance`
`__len1, _Distance __len2, _Compare __comp)`
- `template<typename _Iterator >`
`_Miter_base< _Iterator >::iterator_type __miter_base (_Iterator __it)`
- `template<typename _Iterator >`
`void __move_median_first (_Iterator __a, _Iterator __b, _Iterator __c)`
- `template<typename _Iterator , typename _Compare >`
`void __move_median_first (_Iterator __a, _Iterator __b, _Iterator __c, _Compare`
`__comp)`
- `template<typename _Iterator >`
`_Niter_base< _Iterator >::iterator_type __niter_base (_Iterator __it)`
- `void __once_proxy ()`
- `template<typename _CharT , typename _Traits >`
`void __ostream_fill (basic_ostream< _CharT, _Traits > &__out, streamsize _-`
`__n)`
- `template<typename _CharT , typename _Traits >`
`basic_ostream< _CharT, _Traits > & __ostream_insert (basic_ostream< _-`
`CharT, _Traits > &__out, const _CharT *__s, streamsize __n)`
- `template ostream & __ostream_insert (ostream &, const char *, streamsize)`
- `template wostream & __ostream_insert (wostream &, const wchar_t *, stream-`
`size)`
- `template<typename _CharT , typename _Traits >`
`void __ostream_write (basic_ostream< _CharT, _Traits > &__out, const _-`
`CharT *__s, streamsize __n)`
- `template<typename _ForwardIterator , typename _Predicate >`
`_ForwardIterator __partition (_ForwardIterator __first, _ForwardIterator __last,`
`_Predicate __pred, forward_iterator_tag)`

- `template<typename _BidirectionalIterator, typename _Predicate >`
`_BidirectionalIterator __partition (_BidirectionalIterator __first, _-`
`BidirectionalIterator __last, _Predicate __pred, bidirectional_iterator_tag)`
- `template<typename _RandomAccessIterator >`
`void __pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`
`last, _RandomAccessIterator __result)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void __pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`
`last, _RandomAccessIterator __result, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp >`
`void __push_heap (_RandomAccessIterator __first, _Distance __holeIndex, _-`
`Distance __topIndex, _Tp __value)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _-`
`Compare >`
`void __push_heap (_RandomAccessIterator __first, _Distance __holeIndex, _-`
`Distance __topIndex, _Tp __value, _Compare __comp)`
- `template<typename _BidirectionalIterator >`
`void __reverse (_BidirectionalIterator __first, _BidirectionalIterator __last,`
`bidirectional_iterator_tag)`
- `template<typename _RandomAccessIterator >`
`void __reverse (_RandomAccessIterator __first, _RandomAccessIterator __last,`
`random_access_iterator_tag)`
- `template<typename _ForwardIterator >`
`void __rotate (_ForwardIterator __first, _ForwardIterator __middle, _-`
`ForwardIterator __last, forward_iterator_tag)`
- `template<typename _BidirectionalIterator >`
`void __rotate (_BidirectionalIterator __first, _BidirectionalIterator __middle, _-`
`BidirectionalIterator __last, bidirectional_iterator_tag)`
- `template<typename _RandomAccessIterator >`
`void __rotate (_RandomAccessIterator __first, _RandomAccessIterator __-`
`middle, _RandomAccessIterator __last, random_access_iterator_tag)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _-`
`Distance >`
`_BidirectionalIterator1 __rotate_adaptive (_BidirectionalIterator1 __first, _-`
`BidirectionalIterator1 __middle, _BidirectionalIterator1 __last, _Distance __-`
`len1, _Distance __len2, _BidirectionalIterator2 __buffer, _Distance __buffer_-`
`size)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp >`
`_ForwardIterator __search_n (_ForwardIterator __first, _ForwardIterator __last,`
`_Integer __count, const _Tp &__val, std::forward_iterator_tag)`
- `template<typename _RandomAccessIter, typename _Integer, typename _Tp >`
`_RandomAccessIter __search_n (_RandomAccessIter __first, _-`
`RandomAccessIter __last, _Integer __count, const _Tp &__val, std::random_-`
`access_iterator_tag)`

- `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`
`_ForwardIterator __search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, std::forward_iterator_tag)`
- `template<typename _RandomAccessIter, typename _Integer, typename _Tp, typename _BinaryPredicate >`
`_RandomAccessIter __search_n (_RandomAccessIter __first, _RandomAccessIter __last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, std::random_access_iterator_tag)`
- `void __set_once_functor_lock_ptr (unique_lock< mutex > *)`
- `template<typename _ForwardIterator, typename _Pointer, typename _Predicate, typename _Distance >`
`_ForwardIterator __stable_partition_adaptive (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, _Distance __len, _Pointer __buffer, _Distance __buffer_size)`
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Distance >`
`void __stable_sort_adaptive (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer, _Distance __buffer_size)`
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Distance, typename _Compare >`
`void __stable_sort_adaptive (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer, _Distance __buffer_size, _Compare __comp)`
- `void __throw_bad_alloc (void) __attribute__((__noreturn__))`
- `void __throw_bad_cast (void) __attribute__((__noreturn__))`
- `void __throw_bad_exception (void) __attribute__((__noreturn__))`
- `void __throw_bad_function_call () __attribute__((__noreturn__))`
- `void __throw_bad_typeid (void) __attribute__((__noreturn__))`
- `void __throw_bad_weak_ptr ()`
- `void __throw_domain_error (const char *) __attribute__((__noreturn__))`
- `void __throw_future_error (int) __attribute__((__noreturn__))`
- `void __throw_invalid_argument (const char *) __attribute__((__noreturn__))`
- `void __throw_ios_failure (const char *) __attribute__((__noreturn__))`
- `void __throw_length_error (const char *) __attribute__((__noreturn__))`
- `void __throw_logic_error (const char *) __attribute__((__noreturn__))`
- `void __throw_out_of_range (const char *) __attribute__((__noreturn__))`
- `void __throw_overflow_error (const char *) __attribute__((__noreturn__))`
- `void __throw_range_error (const char *) __attribute__((__noreturn__))`
- `void __throw_regex_error (regex_constants::error_type __ecode)`
- `void __throw_runtime_error (const char *) __attribute__((__noreturn__))`
- `void __throw_system_error (int) __attribute__((__noreturn__))`
- `void __throw_underflow_error (const char *) __attribute__((__noreturn__))`

- `template<typename _Ex >`
`void __throw_with_nested (_Ex &&, const nested_exception *=0) __-`
`attribute__((__noreturn__))`
- `template<typename _Ex >`
`void __throw_with_nested (_Ex &&,...) __attribute__((__noreturn__))`
- `template<typename _Lock >`
`unique_lock< _Lock > __try_to_lock (_Lock &__l)`
- `template<typename... _TElements, std::size_t... _TIdx, typename... _UElements, std::size_t... _UIdx>`
`tuple< _TElements..., _UElements...> __tuple_cat_helper (tuple< _-`
`TElements...> &&__t, const __index_holder< _TIdx...> &, const tuple< _`
`UElements...> &__u, const __index_holder< _UIdx...> &)`
- `template<typename... _TElements, std::size_t... _TIdx, typename... _UElements, std::size_t... _UIdx>`
`tuple< _TElements..., _UElements...> __tuple_cat_helper (tuple< _-`
`TElements...> &&__t, const __index_holder< _TIdx...> &, tuple< _-`
`UElements...> &&__u, const __index_holder< _UIdx...> &)`
- `template<typename... _TElements, std::size_t... _TIdx, typename... _UElements, std::size_t... _UIdx>`
`tuple< _TElements..., _UElements...> __tuple_cat_helper (const tuple< _-`
`TElements...> &__t, const __index_holder< _TIdx...> &, const tuple< _-`
`UElements...> &__u, const __index_holder< _UIdx...> &)`
- `template<typename... _TElements, std::size_t... _TIdx, typename... _UElements, std::size_t... _UIdx>`
`tuple< _TElements..., _UElements...> __tuple_cat_helper (const tuple< _-`
`TElements...> &__t, const __index_holder< _TIdx...> &, tuple< _-`
`UElements...> &&__u, const __index_holder< _UIdx...> &)`
- `template<typename _RandomAccessIterator >`
`void __unguarded_insertion_sort (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void __unguarded_insertion_sort (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void __unguarded_linear_insert (_RandomAccessIterator __last, _Compare __-`
`comp)`
- `template<typename _RandomAccessIterator >`
`void __unguarded_linear_insert (_RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Tp >`
`_RandomAccessIterator __unguarded_partition (_RandomAccessIterator __-`
`first, _RandomAccessIterator __last, const _Tp &__pivot)`
- `template<typename _RandomAccessIterator, typename _Tp, typename _Compare >`
`_RandomAccessIterator __unguarded_partition (_RandomAccessIterator __-`
`first, _RandomAccessIterator __last, const _Tp &__pivot, _Compare __comp)`

- `template<typename _RandomAccessIterator, typename _Compare >`
`_RandomAccessIterator __unguarded_partition_pivot (_RandomAccessIterator`
`__first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`_RandomAccessIterator __unguarded_partition_pivot (_RandomAccessIterator`
`__first, _RandomAccessIterator __last)`
- `template<typename _ForwardIterator, typename _Tp >`
`void __uninitialized_construct_buf (_ForwardIterator __first, _-`
`ForwardIterator __last, _Tp &__value)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator >`
`_ForwardIterator __uninitialized_copy_a (_InputIterator __first, _InputIterator`
`__last, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Tp >`
`_ForwardIterator __uninitialized_copy_a (_InputIterator __first, _InputIterator`
`__last, _ForwardIterator __result, allocator<_Tp > &)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _ForwardIterator, type-`
`name _Allocator >`
`_ForwardIterator __uninitialized_copy_move (_InputIterator1 __first1, _-`
`InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _-`
`ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator >`
`_ForwardIterator __uninitialized_copy_n (_InputIterator __first, _Size __n, _-`
`ForwardIterator __result, input_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Size, typename _ForwardIterator >`
`_ForwardIterator __uninitialized_copy_n (_RandomAccessIterator __first, _-`
`Size __n, _ForwardIterator __result, random_access_iterator_tag)`
- `template<typename _ForwardIterator >`
`void __uninitialized_default (_ForwardIterator __first, _ForwardIterator __-`
`last)`
- `template<typename _ForwardIterator, typename _Allocator >`
`void __uninitialized_default_a (_ForwardIterator __first, _ForwardIterator __-`
`last, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Tp >`
`void __uninitialized_default_a (_ForwardIterator __first, _ForwardIterator __-`
`last, allocator<_Tp > &)`
- `template<typename _ForwardIterator, typename _Size >`
`void __uninitialized_default_n (_ForwardIterator __first, _Size __n)`
- `template<typename _ForwardIterator, typename _Size, typename _Allocator >`
`void __uninitialized_default_n_a (_ForwardIterator __first, _Size __n, _-`
`Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp >`
`void __uninitialized_default_n_a (_ForwardIterator __first, _Size __n, alloca-`
`tor<_Tp > &)`

- `template<typename _ForwardIterator, typename _Tp, typename _Allocator >`
`void __uninitialized_fill_a (_ForwardIterator __first, _ForwardIterator __last,`
`const _Tp &__x, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Tp, typename _Tp2 >`
`void __uninitialized_fill_a (_ForwardIterator __first, _ForwardIterator __last,`
`const _Tp &__x, allocator< _Tp2 > &)`
- `template<typename _ForwardIterator, typename _Tp, typename _InputIterator, typename _`
`Allocator >`
`_ForwardIterator __uninitialized_fill_move (_ForwardIterator __result, _`
`ForwardIterator __mid, const _Tp &__x, _InputIterator __first, _InputIterator`
`__last, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp, typename _Allocator >`
`void __uninitialized_fill_n_a (_ForwardIterator __first, _Size __n, const _Tp`
`&__x, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp, typename _Tp2 >`
`void __uninitialized_fill_n_a (_ForwardIterator __first, _Size __n, const _Tp`
`&__x, allocator< _Tp2 > &)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator >`
`_ForwardIterator __uninitialized_move_a (_InputIterator __first, _InputIterator`
`__last, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _ForwardIterator, type-`
`name _Allocator >`
`_ForwardIterator __uninitialized_move_copy (_InputIterator1 __first1, _`
`InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _`
`ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Tp, typename _`
`Allocator >`
`void __uninitialized_move_fill (_InputIterator __first1, _InputIterator __last1,`
`_ForwardIterator __first2, _ForwardIterator __last2, const _Tp &__x, _Allocator`
`&__alloc)`
- `template<typename _InputIterator, typename _OutputIterator >`
`_OutputIterator __unique_copy (_InputIterator __first, _InputIterator __last, _`
`OutputIterator __result, input_iterator_tag, output_iterator_tag)`
- `template<typename _InputIterator, typename _ForwardIterator >`
`_ForwardIterator __unique_copy (_InputIterator __first, _InputIterator __last, _`
`ForwardIterator __result, input_iterator_tag, forward_iterator_tag)`
- `template<typename _ForwardIterator, typename _OutputIterator >`
`_OutputIterator __unique_copy (_ForwardIterator __first, _ForwardIterator __`
`last, _OutputIterator __result, forward_iterator_tag, output_iterator_tag)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate >`
`_OutputIterator __unique_copy (_InputIterator __first, _InputIterator __last,`
`_OutputIterator __result, _BinaryPredicate __binary_pred, input_iterator_tag,
output_iterator_tag)`

- `template<typename _ForwardIterator, typename _OutputIterator, typename _BinaryPredicate >`
`_OutputIterator __unique_copy (_ForwardIterator __first, _ForwardIterator __-`
`__last, _OutputIterator __result, _BinaryPredicate __binary_pred, forward-`
`iterator_tag, output_iterator_tag)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`
`_ForwardIterator __unique_copy (_InputIterator __first, _InputIterator __last,`
`_ForwardIterator __result, _BinaryPredicate __binary_pred, input_iterator_tag,`
`forward_iterator_tag)`
- `template<typename _Bi_iter >`
`const sub_match< _Bi_iter > & __unmatched_sub ()`
- `template<typename _Tp >`
`void __valarray_copy (const _Tp *__restrict __a, size_t __n, _Tp *__-`
`restrict __b)`
- `template<typename _Tp >`
`void __valarray_copy (const _Tp *__restrict __a, size_t __n, size_t __s, _Tp`
`*__restrict __b)`
- `template<typename _Tp >`
`void __valarray_copy (const _Tp *__restrict __a, _Tp *__restrict __b,`
`size_t __n, size_t __s)`
- `template<typename _Tp >`
`void __valarray_copy (const _Tp *__restrict __src, size_t __n, size_t __s1,`
`_Tp *__restrict __dst, size_t __s2)`
- `template<typename _Tp >`
`void __valarray_copy (const _Tp *__restrict __a, const size_t *__restrict`
`__i, _Tp *__restrict __b, size_t __n)`
- `template<typename _Tp >`
`void __valarray_copy (const _Tp *__restrict __a, size_t __n, _Tp *__-`
`restrict __b, const size_t *__restrict __i)`
- `template<typename _Tp >`
`void __valarray_copy (const _Tp *__restrict __src, size_t __n, const size_t`
`*__restrict __i, _Tp *__restrict __dst, const size_t *__restrict __j)`
- `template<typename _Tp >`
`void __valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void __valarray_copy (_Array< _Tp > __a, size_t __n, size_t __s, _Array<`
`_Tp > __b)`
- `template<typename _Tp >`
`void __valarray_copy (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n,`
`size_t __s)`
- `template<typename _Tp >`
`void __valarray_copy (_Array< _Tp > __a, size_t __n, size_t __s1, _Array<`
`_Tp > __b, size_t __s2)`
- `template<typename _Tp >`
`void __valarray_copy (_Array< _Tp > __a, _Array< size_t > __i, _Array<`
`_Tp > __b, size_t __n)`

- `template<typename _Tp >`
`void __valarray_copy (_Array< _Tp > __src, size_t __n, _Array< size_t >`
`__i, _Array< _Tp > __dst, _Array< size_t > __j)`
- `template<typename _Tp >`
`void __valarray_copy (_Array< _Tp > __a, _Array< bool > __m, _Array<`
`_Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void __valarray_copy (_Array< _Tp > __a, _Array< bool > __m, size_t __n,`
`_Array< _Tp > __b, _Array< bool > __k)`
- `template<typename _Tp, class _Dom >`
`void __valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array<`
`_Tp > __a)`
- `template<typename _Tp, class _Dom >`
`void __valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array<`
`_Tp > __a, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void __valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array<`
`_Tp > __a, _Array< size_t > __i)`
- `template<typename _Tp >`
`void __valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b,`
`_Array< bool > __m)`
- `template<typename _Tp >`
`void __valarray_copy (_Array< _Tp > __e, _Array< size_t > __f, size_t __n,`
`_Array< _Tp > __a, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`
`void __valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array<`
`_Tp > __a, _Array< bool > __m)`
- `template<typename _Tp >`
`void __valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b,`
`_Array< size_t > __i)`
- `template<typename _Tp >`
`void __valarray_copy_construct (const _Tp *__restrict __a, size_t __n,`
`size_t __s, _Tp *__restrict __o)`
- `template<typename _Tp >`
`void __valarray_copy_construct (const _Tp *__restrict __a, const size_t *__-`
`__restrict __i, _Tp *__restrict __o, size_t __n)`
- `template<typename _Tp >`
`void __valarray_copy_construct (const _Tp *__b, const _Tp *__e, _Tp *__-`
`__restrict __o)`
- `template<typename _Tp >`
`void __valarray_copy_construct (_Array< _Tp > __a, size_t __n, size_t __s,`
`_Array< _Tp > __b)`
- `template<typename _Tp >`
`void __valarray_copy_construct (_Array< _Tp > __a, _Array< size_t > __i,`
`_Array< _Tp > __b, size_t __n)`

- `template<typename _Tp, class _Dom >`
`void __valarray_copy_construct (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a)`
- `template<typename _Tp >`
`void __valarray_copy_construct (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void __valarray_default_construct (_Tp *__b, _Tp *__e)`
- `template<typename _Tp >`
`void __valarray_destroy_elements (_Tp *__b, _Tp *__e)`
- `template<typename _Tp >`
`void __valarray_fill (_Tp *__restrict __a, const size_t *__restrict __i, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`
`void __valarray_fill (_Tp *__restrict __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`
`void __valarray_fill (_Tp *__restrict __a, size_t __n, size_t __s, const _Tp &__t)`
- `template<typename _Tp >`
`void __valarray_fill (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`
`void __valarray_fill (_Array< _Tp > __a, size_t __n, size_t __s, const _Tp &__t)`
- `template<typename _Tp >`
`void __valarray_fill (_Array< _Tp > __a, size_t __n, _Array< bool > __m, const _Tp &__t)`
- `template<typename _Tp >`
`void __valarray_fill (_Array< _Tp > __a, _Array< size_t > __i, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`
`void __valarray_fill_construct (_Tp *__b, _Tp *__e, const _Tp __t)`
- `void * __valarray_get_memory (size_t __n)`
- `template<typename _Tp >`
`_Tp *__restrict __valarray_get_storage (size_t __n)`
- `template<typename _Ta >`
`_Ta::value_type __valarray_max (const _Ta &__a)`
- `template<typename _Ta >`
`_Ta::value_type __valarray_min (const _Ta &__a)`
- `template<typename _Tp >`
`_Tp __valarray_product (const _Tp *__f, const _Tp *__l)`
- `void __valarray_release_memory (void *__p)`
- `template<typename _Tp >`
`_Tp __valarray_sum (const _Tp *__f, const _Tp *__l)`
- `bool __verify_grouping (const char *__grouping, size_t __grouping_size, const string &__grouping_tmp) throw ()`

- `template<size_t _Ind, typename... _Tp>`
`auto __volget (volatile tuple< _Tp...> &__tuple)-> typename tuple_element<`
`_Ind`
- `typename _Tp auto __volget (const volatile tuple< _Tp...> &__tuple)-> type-`
`name tuple_element< _Ind`
- `template<typename _CharT >`
`ostreambuf_iterator< _CharT > __write (ostreambuf_iterator< _CharT > __s,`
`const _CharT *__ws, int __len)`
- `template<typename _CharT, typename _OutIter >`
`_OutIter __write (_OutIter __s, const _CharT *__ws, int __len)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, _Array<`
`size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, -`
`Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, -`
`Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, const Expr<`
`_Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n,`
`size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< _Tp`
`> __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __s,`
`const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, -`
`Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, _Array<`
`size_t > __i, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< bool`
`> __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< bool`
`> __m, const Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp >`
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n,`
`const _Tp &__t)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, const`
`_Tp &__t)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, _`
`Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, const _Expr<`
`_Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n,`
`size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< _Tp >`
`__b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __s, const`
`_Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, _`
`Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< size_t`
`> __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< bool >`
`__m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< bool >`
`__m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< size_t`
`> __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, _`
`Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _`
`Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< bool`
`> __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp, class _Dom>`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, const _Expr<`
`_Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< _Tp`
`> __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom>`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __s,`
`const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< size_t`
`> __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp>`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _`
`Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom>`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< size_t`
`> __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< bool`
`> __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp>`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _`
`Array< _Tp > __b)`
- `template<typename _Tp>`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n,`
`size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp>`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n,`
`const _Tp &__t)`
- `template<typename _Tp>`
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array<`
`_Tp > __b)`
- `template<typename _Tp, class _Dom>`
`void _Array_augmented__divides (_Array< _Tp > __a, const _Expr< _`
`Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, size_t`
`__s, _Array< _Tp > __b)`
- `template<typename _Tp>`
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< _Tp > _`
`__b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom>`
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __s, const _`
`Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp >`
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array<`
`_Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< size_t >`
`__i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< bool > __-`
`_m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, const _-`
`Tp &__t)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< bool > __-`
`_m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< size_t >`
`__i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array<`
`_Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array<`
`_Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, const _Tp`
`&__t)`
- `template<typename _Tp >`
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array<`
`_Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__minus (_Array< _Tp > __a, const _Expr< _Dom,`
`_Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __s, const _-`
`Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< size_t >`
`__i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array<`
`_Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< size_t >`
`__i, const _Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp >`
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< bool > __-`
`m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< bool > __-`
`m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, size_t __-`
`s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< _Tp > __b,`
`size_t __n, size_t __s)`
- `template<typename _Tp >`
`void _Array_augmented__modulus (_Array< _Tp > __a, _Array< size_t >`
`__i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__modulus (_Array< _Tp > __a, const _Expr< _-`
`Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __n, size_t`
`__s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void _Array_augmented__modulus (_Array< _Tp > __a, _Array< _Tp >`
`__b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __s, const`
`_Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__modulus (_Array< _Tp > __a, _Array< size_t >`
`__i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__modulus (_Array< _Tp > __a, _Array< bool >`
`__m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__modulus (_Array< _Tp > __a, _Array< bool >`
`__m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __n, const`
`_Tp &__t)`
- `template<typename _Tp >`
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b, _Array< size_t > __i)`

- `template<typename _Tp >`
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, _Array< size_t`
`> __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, const _Expr< _-`
`Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, _Array< size_t`
`> __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, const`
`_Tp &__t)`
- `template<typename _Tp >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b)`
- `template<typename _Tp >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, size_t`
`__s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, _Array< _Tp >`
`__b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __s, const`
`_Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, _Array< bool >`
`__m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, _Array< bool >`
`__m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, _Array<`
`_Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, const _Tp`
`&__t)`

- `template<typename _Tp >`
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, _Array<`
`_Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__plus (_Array< _Tp > __a, const _Expr< _Dom,`
`_Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, size_t __s,`
`_Array< _Tp > __b)`
- `template<typename _Tp >`
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< _Tp > __b,`
`size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __s, const _`
`Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< size_t > __i,`
`const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< bool > __m,`
`_Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< bool > __m,`
`const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< size_t > __i,`
`_Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, _Array<`
`_Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, _`
`Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, const`
`_Tp &__t)`
- `template<typename _Tp >`
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, _`
`Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__shift_left (_Array< _Tp > __a, const _Expr< _`
`Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, size_t`
`__s, _Array< _Tp > __b)`

- `template<typename _Tp >`
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< _Tp >`
`__b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __s, const`
`_Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, _`
`Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< size_t >`
`__i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< bool >`
`__m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< bool >`
`__m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< size_t >`
`__i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< bool >`
`__m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _`
`Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__shift_right (_Array< _Tp > __a, const _Expr<`
`_Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, const`
`_Tp &__t)`
- `template<typename _Tp >`
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _`
`Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __s, const`
`_Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< size_t`
`> __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _`
`Array< _Tp > __b, _Array< size_t > __i)`

- `template<typename _Tp, class _Dom >`
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< size_t >`
`> __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< bool >`
`__m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n,`
`size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< _Tp >`
`__b, size_t __n, size_t __s)`
- `template<typename _T1, typename... _Args>`
`void _Construct (_T1 *__p, _Args &&... __args)`
- `template<typename _Tp >`
`void _Destroy (_Tp *__pointer)`
- `template<typename _ForwardIterator >`
`void _Destroy (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Tp >`
`void _Destroy (_ForwardIterator __first, _ForwardIterator __last, allocator< _`
`_Tp > &)`
- `template<typename _ForwardIterator, typename _Allocator >`
`void _Destroy (_ForwardIterator __first, _ForwardIterator __last, _Allocator`
`&__alloc)`
- `size_t _Fnv_hash_bytes (const void *__ptr, size_t __len, size_t __seed)`
- `size_t _Hash_bytes (const void *__ptr, size_t __len, size_t __seed)`
- `unsigned int _Rb_tree_black_count (const _Rb_tree_node_base *__node,`
`const _Rb_tree_node_base *__root) throw ()`
- `_Rb_tree_node_base * _Rb_tree_decrement (_Rb_tree_node_base *__x)`
`throw ()`
- `const _Rb_tree_node_base * _Rb_tree_decrement (const _Rb_tree_node_base`
`*__x) throw ()`
- `_Rb_tree_node_base * _Rb_tree_increment (_Rb_tree_node_base *__x) throw`
`()`
- `const _Rb_tree_node_base * _Rb_tree_increment (const _Rb_tree_node_base`
`*__x) throw ()`
- `void _Rb_tree_insert_and_rebalance (const bool __insert_left, _Rb_tree_`
`node_base *__x, _Rb_tree_node_base *__p, _Rb_tree_node_base &__header)`
`throw ()`
- `_Rb_tree_node_base * _Rb_tree_rebalance_for_erase (_Rb_tree_node_base`
`*const __z, _Rb_tree_node_base &__header) throw ()`
- `void abort (void) throw ()`
- `template<typename _Tp >`
`_Tp abs (const complex< _Tp > &)`

- double **abs** (double __x)
- float **abs** (float __x)
- long double **abs** (long double __x)
- template<typename _Tp >
__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **abs**
(_Tp __x)
- template<class _Dom >
_Expr< _UnClos< _Abs, _Expr, _Dom >, typename _Dom::value_type > **abs**
(const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp >
_Expr< _UnClos< _Abs, _ValArray, _Tp >, _Tp > **abs** (const [valarray](#)< _Tp
> &__v)
- template<typename _InputIterator, typename _Tp >
_Tp **accumulate** (_InputIterator __first, _InputIterator __last, _Tp __init)
- template<typename _InputIterator, typename _Tp, typename _BinaryOperation >
_Tp **accumulate** (_InputIterator __first, _InputIterator __last, _Tp __init, _
BinaryOperation __binary_op)
- template<typename _Tp >
[std::complex](#)< _Tp > **acos** (const [std::complex](#)< _Tp > &__z)
- float **acos** (float __x)
- long double **acos** (long double __x)
- template<typename _Tp >
__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type
acos (_Tp __x)
- template<class _Dom >
_Expr< _UnClos< _Acos, _Expr, _Dom >, typename _Dom::value_type >
acos (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp >
_Expr< _UnClos< _Acos, _ValArray, _Tp >, _Tp > **acos** (const [valarray](#)< _Tp
> &__v)
- template<typename _Tp >
[std::complex](#)< _Tp > **acosh** (const [std::complex](#)< _Tp > &__z)
- template<typename _Tp >
_Tp * **addressof** (_Tp &__r)
- template<typename _InputIterator, typename _OutputIterator >
_OutputIterator **adjacent_difference** (_InputIterator __first, _InputIterator __last,
_OutputIterator __result)
- template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation >
_OutputIterator **adjacent_difference** (_InputIterator __first, _InputIterator __last,
_OutputIterator __result, _BinaryOperation __binary_op)
- template<typename _FIter >
_FIter **adjacent_find** (_FIter, _FIter)
- template<typename _FIter, typename _BinaryPredicate >
_FIter **adjacent_find** (_FIter, _FIter, _BinaryPredicate)

- `template<typename _ForwardIterator >`
`_ForwardIterator adjacent_find (_ForwardIterator __first, _ForwardIterator __-`
`last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`
`_ForwardIterator adjacent_find (_ForwardIterator __first, _ForwardIterator __-`
`last, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Distance >`
`void advance (_InputIterator &__i, _Distance __n)`
- `template<typename _Iter, typename _Predicate >`
`bool all_of (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`
`bool all_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Tp, typename _Alloc, typename... _Args>`
`shared_ptr< _Tp > allocate_shared (const _Alloc &__a, _Args &&...__args)`
- `template<typename _Iter, typename _Predicate >`
`bool any_of (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`
`bool any_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type arg (_Tp __x)`
- `template<typename _Tp >`
`_Tp arg (const complex< _Tp > &)`
- `template<typename _Tp >`
`std::complex< _Tp > asin (const std::complex< _Tp > &__z)`
- `float asin (float __x)`
- `long double asin (long double __x)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`
`asin (_Tp __x)`
- `template<class _Dom >`
`_Expr< _UnClos< _Asin, _Expr, _Dom >, typename _Dom::value_type > asin`
`(const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Asin, _ValArray, _Tp >, _Tp > asin (const valarray< _Tp`
`> &__v)`
- `template<typename _Tp >`
`std::complex< _Tp > asinh (const std::complex< _Tp > &__z)`
- `template<typename _Fn, typename... _Args>`
`future< typename result_of< _Fn(_Args...)>::type > async (launch __policy,`
`_Fn &&__fn, _Args &&...__args)`
- `template<typename _Fn, typename... _Args>`
`enable_if< !is_same< typename decay< _Fn >::type, launch >::value, future<`
`decltype(std::declval< _Fn >)(std::declval< _Args >)...> >::type async (_Fn`
`&&__fn, _Args &&...__args)`

- `template<typename _Tp >`
`std::complex<_Tp > atan (const std::complex<_Tp > &__z)`
- `float atan (float __x)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer<_Tp >::__value, double >::__type`
`atan (_Tp __x)`
- `long double atan (long double __x)`
- `template<class _Dom >`
`_Expr< _UnClos< _Atan, _Expr, _Dom >, typename _Dom::value_type >`
`atan (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Atan, _ValArray, _Tp >, _Tp > atan (const valarray<_Tp`
`> &__v)`
- `float atan2 (float __y, float __x)`
- `long double atan2 (long double __y, long double __x)`
- `template<typename _Tp, typename _Up >`
`__gnu_cxx::__promote_2< typename __gnu_cxx::__enable_if< __is-`
`arithmetic<_Tp >::__value &&__is_arithmetic<_Up >::__value, _Tp`
`>::__type, _Up >::__type atan2 (_Tp __y, _Up __x)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< _Atan2, _Expr, _Expr, _Dom1, _Dom2 >, typename _-`
`Dom1::value_type > atan2 (const _Expr< _Dom1, typename _Dom1::value-`
`type > &__e1, const _Expr< _Dom2, typename _Dom2::value_type > &__e2)`
- `template<class _Dom >`
`_Expr< _BinClos< _Atan2, _Expr, _ValArray, _Dom, typename _-`
`Dom::value_type >, typename _Dom::value_type > atan2 (const _Expr< _-`
`Dom, typename _Dom::value_type > &__e, const valarray< typename _-`
`Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< _Atan2, _ValArray, _Expr, typename _Dom::value_type,`
`_Dom >, typename _Dom::value_type > atan2 (const valarray< typename _-`
`Dom::valarray > &__v, const _Expr< _Dom, typename _Dom::value_type >`
`&__e)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Atan2, _ValArray, _ValArray, _Tp, _Tp >, _Tp > atan2`
`(const valarray<_Tp > &__v, const valarray<_Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Atan2, _ValArray, _Constant, _Tp, _Tp >, _Tp > atan2`
`(const valarray<_Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Atan2, _Constant, _ValArray, _Tp, _Tp >, _Tp > atan2`
`(const _Tp &__t, const valarray<_Tp > &__v)`

- `template<class _Dom >`
`_Expr< _BinClos< _Atan2, _Expr, _Constant, _Dom, typename _Dom::value_`
`type >, typename _Dom::value_type > atan2 (const _Expr< _Dom, typename`
`_Dom::value_type > &__e, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< _Atan2, _Constant, _Expr, typename _Dom::value_type, _`
`Dom >, typename _Dom::value_type > atan2 (const typename _Dom::value_`
`type &__t, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`std::complex< _Tp > atanh (const std::complex< _Tp > &__z)`
- `int atexit (void(*)()) throw ()`
- `bool atomic_compare_exchange_strong (atomic_bool *__a, bool *__i1, bool`
`__i2)`
- `bool atomic_compare_exchange_strong (volatile atomic_bool *__a, bool *__`
`i1, bool __i2)`
- `template<typename _ITp >`
`bool atomic_compare_exchange_strong (volatile __atomic_base< _ITp > *_`
`_a, _ITp *__i1, _ITp __i2)`
- `bool atomic_compare_exchange_strong (atomic_address *__a, void **__v1,`
`void *__v2)`
- `bool atomic_compare_exchange_strong (volatile atomic_address *__a, void`
`**__v1, void *__v2)`
- `template<typename _ITp >`
`bool atomic_compare_exchange_strong (__atomic_base< _ITp > *__a, _ITp`
`*__i1, _ITp __i2)`
- `bool atomic_compare_exchange_strong_explicit (volatile atomic_bool *__a,`
`bool *__i1, bool __i2, memory_order __m1, memory_order __m2)`
- `bool atomic_compare_exchange_strong_explicit (atomic_bool *__a, bool *__`
`i1, bool __i2, memory_order __m1, memory_order __m2)`
- `bool atomic_compare_exchange_strong_explicit (volatile atomic_address *__`
`a, void **__v1, void *__v2, memory_order __m1, memory_order __m2)`
- `template<typename _ITp >`
`bool atomic_compare_exchange_strong_explicit (__atomic_base< _ITp >`
`*__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2)`
- `template<typename _ITp >`
`bool atomic_compare_exchange_strong_explicit (volatile __atomic_base< _`
`ITp > *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2)`
- `bool atomic_compare_exchange_strong_explicit (atomic_address *__a, void`
`**__v1, void *__v2, memory_order __m1, memory_order __m2)`
- `bool atomic_compare_exchange_weak (atomic_bool *__a, bool *__i1, bool`
`__i2)`
- `bool atomic_compare_exchange_weak (volatile atomic_bool *__a, bool *__`
`i1, bool __i2)`

- **bool atomic_compare_exchange_weak** (atomic_address *__a, void **__v1, void *__v2)
- **bool atomic_compare_exchange_weak** (volatile atomic_address *__a, void **__v1, void *__v2)
- **template<typename ITp >**
bool atomic_compare_exchange_weak (volatile __atomic_base< ITp > *__a, ITp *__i1, ITp __i2)
- **template<typename ITp >**
bool atomic_compare_exchange_weak (__atomic_base< ITp > *__a, ITp *__i1, ITp __i2)
- **bool atomic_compare_exchange_weak_explicit** (volatile [atomic_bool](#) *__a, bool *__i1, bool __i2, [memory_order](#) __m1, [memory_order](#) __m2)
- **bool atomic_compare_exchange_weak_explicit** ([atomic_bool](#) *__a, bool *__i1, bool __i2, [memory_order](#) __m1, [memory_order](#) __m2)
- **template<typename ITp >**
bool atomic_compare_exchange_weak_explicit (__atomic_base< ITp > *__a, ITp *__i1, ITp __i2, [memory_order](#) __m1, [memory_order](#) __m2)
- **template<typename ITp >**
bool atomic_compare_exchange_weak_explicit (volatile __atomic_base< ITp > *__a, ITp *__i1, ITp __i2, [memory_order](#) __m1, [memory_order](#) __m2)
- **bool atomic_compare_exchange_weak_explicit** (volatile atomic_address *__a, void **__v1, void *__v2, [memory_order](#) __m1, [memory_order](#) __m2)
- **bool atomic_compare_exchange_weak_explicit** (atomic_address *__a, void **__v1, void *__v2, [memory_order](#) __m1, [memory_order](#) __m2)
- **bool atomic_exchange** ([atomic_bool](#) *__a, bool __i)
- **bool atomic_exchange** (volatile [atomic_bool](#) *__a, bool __i)
- **void * atomic_exchange** (volatile atomic_address *__a, void *__v)
- **void * atomic_exchange** (atomic_address *__a, void *__v)
- **template<typename ITp >**
ITp atomic_exchange (__atomic_base< ITp > *__a, ITp __i)
- **template<typename ITp >**
ITp atomic_exchange (volatile __atomic_base< ITp > *__a, ITp __i)
- **bool atomic_exchange_explicit** (volatile [atomic_bool](#) *__a, bool __i, [memory_order](#) __m)
- **bool atomic_exchange_explicit** ([atomic_bool](#) *__a, bool __i, [memory_order](#) __m)
- **void * atomic_exchange_explicit** (atomic_address *__a, void *__v, [memory_order](#) __m)
- **void * atomic_exchange_explicit** (volatile atomic_address *__a, void *__v, [memory_order](#) __m)
- **template<typename ITp >**
ITp atomic_exchange_explicit (__atomic_base< ITp > *__a, ITp __i, [memory_order](#) __m)

- `template<typename _ITp >`
`_ITp atomic_exchange_explicit (volatile __atomic_base< _ITp > *__a, _ITp`
`__i, memory_order __m)`
- `void * atomic_fetch_add (atomic_address *__a, ptrdiff_t __d)`
- `template<typename _ITp >`
`_ITp atomic_fetch_add (__atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`
`_ITp atomic_fetch_add (volatile __atomic_base< _ITp > *__a, _ITp __i)`
- `void * atomic_fetch_add (volatile atomic_address *__a, ptrdiff_t __d)`
- `void * atomic_fetch_add_explicit (volatile atomic_address *__a, ptrdiff_t __d,`
`memory_order __m)`
- `void * atomic_fetch_add_explicit (atomic_address *__a, ptrdiff_t __d,`
`memory_order __m)`
- `template<typename _ITp >`
`_ITp atomic_fetch_add_explicit (volatile __atomic_base< _ITp > *__a, _ITp`
`__i, memory_order __m)`
- `template<typename _ITp >`
`_ITp atomic_fetch_add_explicit (__atomic_base< _ITp > *__a, _ITp __i,`
`memory_order __m)`
- `template<typename _ITp >`
`_ITp atomic_fetch_and (volatile __atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`
`_ITp atomic_fetch_and (__atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`
`_ITp atomic_fetch_and_explicit (volatile __atomic_base< _ITp > *__a, _ITp`
`__i, memory_order __m)`
- `template<typename _ITp >`
`_ITp atomic_fetch_and_explicit (__atomic_base< _ITp > *__a, _ITp __i,`
`memory_order __m)`
- `template<typename _ITp >`
`_ITp atomic_fetch_or (__atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`
`_ITp atomic_fetch_or (volatile __atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`
`_ITp atomic_fetch_or_explicit (__atomic_base< _ITp > *__a, _ITp __i,`
`memory_order __m)`
- `template<typename _ITp >`
`_ITp atomic_fetch_or_explicit (volatile __atomic_base< _ITp > *__a, _ITp`
`__i, memory_order __m)`
- `void * atomic_fetch_sub (volatile atomic_address *__a, ptrdiff_t __d)`
- `void * atomic_fetch_sub (atomic_address *__a, ptrdiff_t __d)`
- `template<typename _ITp >`
`_ITp atomic_fetch_sub (__atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`
`_ITp atomic_fetch_sub (volatile __atomic_base< _ITp > *__a, _ITp __i)`

- `template<typename _ITp >`
`_ITp atomic_fetch_sub_explicit (volatile __atomic_base< _ITp > *__a, _ITp __i, memory_order __m)`
- `template<typename _ITp >`
`_ITp atomic_fetch_sub_explicit (__atomic_base< _ITp > *__a, _ITp __i, memory_order __m)`
- `void * atomic_fetch_sub_explicit (volatile atomic_address *__a, ptrdiff_t __d, memory_order __m)`
- `void * atomic_fetch_sub_explicit (atomic_address *__a, ptrdiff_t __d, memory_order __m)`
- `template<typename _ITp >`
`_ITp atomic_fetch_xor (volatile __atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`
`_ITp atomic_fetch_xor (__atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`
`_ITp atomic_fetch_xor_explicit (volatile __atomic_base< _ITp > *__a, _ITp __i, __i, memory_order __m)`
- `template<typename _ITp >`
`_ITp atomic_fetch_xor_explicit (__atomic_base< _ITp > *__a, _ITp __i, memory_order __m)`
- `void atomic_flag_clear (atomic_flag *__a)`
- `void atomic_flag_clear (volatile atomic_flag *__a)`
- `void atomic_flag_clear_explicit (volatile atomic_flag *__a, memory_order __m)`
- `void atomic_flag_clear_explicit (atomic_flag *__a, memory_order __m)`
- `bool atomic_flag_test_and_set (atomic_flag *__a)`
- `bool atomic_flag_test_and_set (volatile atomic_flag *__a)`
- `bool atomic_flag_test_and_set_explicit (volatile atomic_flag *__a, memory_order __m)`
- `bool atomic_flag_test_and_set_explicit (atomic_flag *__a, memory_order __m)`
- `template<typename _ITp >`
`void atomic_init (__atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`
`void atomic_init (volatile __atomic_base< _ITp > *__a, _ITp __i)`
- `void atomic_init (atomic_address *__a, void *__v)`
- `void atomic_init (volatile atomic_address *__a, void *__v)`
- `void atomic_init (volatile atomic_bool *__a, bool __b)`
- `void atomic_init (atomic_bool *__a, bool __b)`
- `template<typename _ITp >`
`bool atomic_is_lock_free (const __atomic_base< _ITp > *__a)`
- `template<typename _ITp >`
`bool atomic_is_lock_free (const volatile __atomic_base< _ITp > *__a)`
- `bool atomic_is_lock_free (const atomic_address *__a)`

- `bool atomic_is_lock_free (const volatile atomic_address *__a)`
- `bool atomic_is_lock_free (const atomic_bool *__a)`
- `bool atomic_is_lock_free (const volatile atomic_bool *__a)`
- `bool atomic_load (const atomic_bool *__a)`
- `bool atomic_load (const volatile atomic_bool *__a)`
- `void * atomic_load (const atomic_address *__a)`
- `void * atomic_load (const volatile atomic_address *__a)`
- `template<typename _ITp >
_ITp atomic_load (const __atomic_base< _ITp > *__a)`
- `template<typename _ITp >
_ITp atomic_load (const volatile __atomic_base< _ITp > *__a)`
- `bool atomic_load_explicit (const atomic_bool *__a, memory_order __m)`
- `bool atomic_load_explicit (const volatile atomic_bool *__a, memory_order __m)`
- `void * atomic_load_explicit (const volatile atomic_address *__a, memory_order __m)`
- `void * atomic_load_explicit (const atomic_address *__a, memory_order __m)`
- `template<typename _ITp >
_ITp atomic_load_explicit (const __atomic_base< _ITp > *__a, memory_order __m)`
- `template<typename _ITp >
_ITp atomic_load_explicit (const volatile __atomic_base< _ITp > *__a, memory_order __m)`
- `void atomic_store (volatile atomic_bool *__a, bool __i)`
- `void atomic_store (atomic_bool *__a, bool __i)`
- `void atomic_store (atomic_address *__a, void *__v)`
- `void atomic_store (volatile atomic_address *__a, void *__v)`
- `template<typename _ITp >
void atomic_store (__atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >
void atomic_store (volatile __atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >
void atomic_store_explicit (__atomic_base< _ITp > *__a, _ITp __i, memory_order __m)`
- `template<typename _ITp >
void atomic_store_explicit (volatile __atomic_base< _ITp > *__a, _ITp __i, memory_order __m)`
- `void atomic_store_explicit (atomic_address *__a, void *__v, memory_order __m)`
- `void atomic_store_explicit (atomic_bool *__a, bool __i, memory_order __m)`
- `void atomic_store_explicit (volatile atomic_address *__a, void *__v, memory_order __m)`
- `void atomic_store_explicit (volatile atomic_bool *__a, bool __i, memory_order __m)`

- `template<typename _Container >`
`back_insert_iterator< _Container > back_inserter (_Container &__x)`
- `template<class _Container >`
`auto begin (const _Container &__cont)-> decltype(__cont.begin())`
- `template<class _Tp >`
`constexpr const _Tp * begin (initializer_list< _Tp > __ils)`
- `template<class _Tp >`
`_Tp * begin (valarray< _Tp > &__va)`
- `template<class _Tp >`
`const _Tp * begin (const valarray< _Tp > &__va)`
- `template<class _Container >`
`auto begin (_Container &__cont)-> decltype(__cont.begin())`
- `template<class _Tp , size_t _Nm >`
`_Tp * begin (_Tp(&__arr)[_Nm])`
- `template<typename _ForwardIterator , typename _Tp , typename _Compare >`
`bool binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _Filter , typename _Tp >`
`bool binary_search (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter , typename _Tp , typename _Compare >`
`bool binary_search (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _ForwardIterator , typename _Tp >`
`bool binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _Functor , typename... _ArgTypes >`
`_Bind_helper< _Functor, _ArgTypes...>::type bind (_Functor &&__f, -
_ArgTypes &&...__args)`
- `template<typename _Result , typename _Functor , typename... _ArgTypes >`
`_Bindres_helper< _Result, _Functor, _ArgTypes...>::type bind (_Functor &&-
__f, _ArgTypes &&...__args)`
- `template<typename _Operation , typename _Tp >`
`binder1st< _Operation > bind1st (const _Operation &__fn, const _Tp &__x)`
- `template<typename _Operation , typename _Tp >`
`binder2nd< _Operation > bind2nd (const _Operation &__fn, const _Tp &__x)`
- `ios_base & boolalpha (ios_base &__base)`
- `template<typename _Callable , typename... _Args >`
`void call_once (once_flag &__once, _Callable &&__f, _Args &&...__args)`
- `float ceil (float __x)`
- `long double ceil (long double __x)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type ceil
(_Tp __x)`
- `template<typename _Tp >`
`complex< _Tp > conj (const complex< _Tp > &)`

- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type conj (_Tp __x)`
- `template<typename _Tp, typename _Tp1 >`
`shared_ptr< _Tp > const_pointer_cast (const shared_ptr< _Tp1 > &__r)`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>`
`__shared_ptr< _Tp, _Lp > const_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r)`
- `template<typename _II, typename _OI >`
`_OI copy (_II __first, _II __last, _OI __result)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * > copy (_Deque_iterator< _Tp, _Tp &, _Tp * > __first, _Deque_iterator< _Tp, _Tp &, _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * > copy (_Deque_iterator< _Tp, const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const _Tp &, const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type copy (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, ostreambuf_iterator< _CharT > __result)`
- `template<typename _Iter, typename _OIter >`
`_OIter copy (_Iter, _Iter, _OIter)`
- `template<typename _BI1, typename _BI2 >`
`_BI2 copy_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * > copy_backward (_Deque_iterator< _Tp, _Tp &, _Tp * > __first, _Deque_iterator< _Tp, _Tp &, _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * > copy_backward (_Deque_iterator< _Tp, const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const _Tp &, const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _BIter1, typename _BIter2 >`
`_BIter2 copy_backward (_BIter1, _BIter1, _BIter2)`
- `template<typename _Ex >`
`exception_ptr copy_exception (_Ex __ex) throw ()`
- `template<typename _Iter, typename _OIter, typename _Predicate >`
`_OIter copy_if (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`
`_OutputIterator copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _Iter, typename _Size, typename _OIter >`
`_OIter copy_n (_Iter, _Size, _OIter)`

- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`
`_OutputIterator copy_n (_InputIterator __first, _Size __n, _OutputIterator __-`
`result)`
- `template<typename _Tp >`
`complex<_Tp> cos (const complex<_Tp> &)`
- `float cos (float __x)`
- `long double cos (long double __x)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if<__is_integer<_Tp>::__value, double>::__type cos`
`(_Tp __x)`
- `template<typename _Tp >`
`_Expr<_UnClos<_Cos, _ValArray, _Tp>, _Tp> cos (const valarray<_Tp`
`> &__v)`
- `template<class _Dom >`
`_Expr<_UnClos<_Cos, _Expr, _Dom>, typename _Dom::value_type> cos`
`(const _Expr<_Dom, typename _Dom::value_type> &__e)`
- `template<typename _Tp >`
`complex<_Tp> cosh (const complex<_Tp> &)`
- `float cosh (float __x)`
- `long double cosh (long double __x)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if<__is_integer<_Tp>::__value, double>::__type`
`cosh (_Tp __x)`
- `template<class _Dom >`
`_Expr<_UnClos<_Cosh, _Expr, _Dom>, typename _Dom::value_type> >`
`cosh (const _Expr<_Dom, typename _Dom::value_type> &__e)`
- `template<typename _Tp >`
`_Expr<_UnClos<_Cosh, _ValArray, _Tp>, _Tp> cosh (const valarray<_Tp`
`> &__v)`
- `template<typename _Iter, typename _Tp >`
`iterator_traits<_Iter>::difference_type count (_Iter, _Iter, const _Tp &)`
- `template<typename _InputIterator, typename _Tp >`
`iterator_traits<_InputIterator>::difference_type count (_InputIterator __first,`
`_InputIterator __last, const _Tp &__value)`
- `template<typename _Iter, typename _Predicate >`
`iterator_traits<_Iter>::difference_type count_if (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`
`iterator_traits<_InputIterator>::difference_type count_if (_InputIterator __-`
`first, _InputIterator __last, _Predicate __pred)`
- `exception_ptr current_exception () throw ()`
- `ios_base & dec (ios_base &__base)`
- `typedef decltype (nullptr) nullptr_t`
- `template<typename _Tp >`
`add_rvalue_reference<_Tp>::type declval () noexcept`

- `template<typename _InputIterator >`
`iterator_traits< _InputIterator >::difference_type distance (_InputIterator __-`
`first, _InputIterator __last)`
- `template<typename _Tp, typename _Tp1 >`
`shared_ptr< _Tp > dynamic_pointer_cast (const shared_ptr< _Tp1 > &__r)`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>`
`__shared_ptr< _Tp, _Lp > dynamic_pointer_cast (const __shared_ptr< _Tp1,`
`_Lp > &__r)`
- `template<class _Tp >`
`constexpr const _Tp * end (initializer_list< _Tp > __ils)`
- `template<class _Tp >`
`_Tp * end (valarray< _Tp > &__va)`
- `template<class _Tp >`
`const _Tp * end (const valarray< _Tp > &__va)`
- `template<class _Container >`
`auto end (const _Container &__cont)-> decltype(__cont.end())`
- `template<class _Container >`
`auto end (_Container &__cont)-> decltype(__cont.end())`
- `template<class _Tp, size_t _Nm>`
`_Tp * end (_Tp(&__arr)[_Nm])`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & endl (basic_ostream< _CharT, _Traits >`
`&__os)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & ends (basic_ostream< _CharT, _Traits >`
`&__os)`
- `template<typename _Iter1, typename _Iter2 >`
`bool equal (_Iter1, _Iter1, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`
`bool equal (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _BinaryPredicate`
`__binary_pred)`
- `template<typename _II1, typename _II2 >`
`bool equal (_II1 __first1, _II1 __last1, _II2 __first2)`
- `template<typename _Filter, typename _Tp >`
`pair< _Filter, _Filter > equal_range (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare >`
`pair< _Filter, _Filter > equal_range (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _ForwardIterator, typename _Tp >`
`pair< _ForwardIterator, _ForwardIterator > equal_range (_ForwardIterator __-`
`first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`pair< _ForwardIterator, _ForwardIterator > equal_range (_ForwardIterator __-`
`first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `void exit (int) throw ()`

- template<typename _Tp >
 complex<_Tp> **exp** (const **complex**<_Tp> &)
- float **exp** (float __x)
- long double **exp** (long double __x)
- template<typename _Tp >
 __gnu_cxx::__enable_if< __is_integer<_Tp>::__value, double>::__type **exp**
 (_Tp __x)
- template<class _Dom >
 _Expr<_UnClos<_Exp, _Expr, _Dom>, typename _Dom::value_type> **exp**
 (const _Expr<_Dom, typename _Dom::value_type> &__e)
- template<typename _Tp >
 _Expr<_UnClos<_Exp, _ValArray, _Tp>, _Tp> **exp** (const **valarray**<_Tp>
 &__v)
- template<typename _Tp >
 _Tp **fabs** (const **std::complex**<_Tp> &__z)
- template<typename _Tp >
 __gnu_cxx::__enable_if< __is_integer<_Tp>::__value, double>::__type
 fabs (_Tp __x)
- float **fabs** (float __x)
- long double **fabs** (long double __x)
- template<typename _ForwardIterator, typename _Tp >
 void **fill** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)
- void **fill** (_Bit_iterator __first, _Bit_iterator __last, const bool &__x)
- template<typename _Tp >
 void **fill** (const **_Deque_iterator**<_Tp, _Tp &, _Tp*> &__first, const **_Deque_-**
 iterator<_Tp, _Tp &, _Tp*> &__last, const _Tp &__value)
- template<typename _FIter, typename _Tp >
 void **fill** (_FIter, _FIter, const _Tp &)
- template<typename _OI, typename _Size, typename _Tp >
 _OI **fill_n** (_OI __first, _Size __n, const _Tp &__value)
- template<typename _OIter, typename _Size, typename _Tp >
 _OIter **fill_n** (_OIter, _Size, const _Tp &)
- template<typename _Iter, typename _Tp >
 _Iter **find** (_Iter, _Iter, const _Tp &)
- template<typename _InputIterator, typename _Tp >
 _InputIterator **find** (_InputIterator __first, _InputIterator __last, const _Tp &__-
 val)
- template<typename _CharT >
 __gnu_cxx::__enable_if< __is_char<_CharT>::__value, **istreambuf_-**
 iterator<_CharT> >::__type **find** (**istreambuf_iterator**<_CharT> __first,
 istreambuf_iterator<_CharT> __last, const _CharT &__val)
- template<typename _Filter1, typename _Filter2 >
 _Filter1 **find_end** (_Filter1, _Filter1, _Filter2, _Filter2)
- template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >
 _Filter1 **find_end** (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)

- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`_ForwardIterator1 find_end (_ForwardIterator1 __first1, _ForwardIterator1 __-`
`last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate`
`>`
`_ForwardIterator1 find_end (_ForwardIterator1 __first1, _ForwardIterator1 __-`
`last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate`
`__comp)`
- `template<typename _FIter1, typename _FIter2 >`
`_FIter1 find_first_of (_FIter1, _FIter1, _FIter2, _FIter2)`
- `template<typename _FIter1, typename _FIter2, typename _BinaryPredicate >`
`_FIter1 find_first_of (_FIter1, _FIter1, _FIter2, _FIter2, _BinaryPredicate)`
- `template<typename _InputIterator, typename _ForwardIterator >`
`_InputIterator find_first_of (_InputIterator __first1, _InputIterator __last1, _-`
`ForwardIterator __first2, _ForwardIterator __last2)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`
`_InputIterator find_first_of (_InputIterator __first1, _InputIterator __last1, _-`
`ForwardIterator __first2, _ForwardIterator __last2, _BinaryPredicate __comp)`
- `template<typename _Iter, typename _Predicate >`
`_Iter find_if (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`
`_InputIterator find_if (_InputIterator __first, _InputIterator __last, _Predicate _-`
`__pred)`
- `template<typename _Iter, typename _Predicate >`
`_Iter find_if_not (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`
`_InputIterator find_if_not (_InputIterator __first, _InputIterator __last, _-`
`Predicate __pred)`
- `ios_base & fixed (ios_base & __base)`
- `float floor (float __x)`
- `long double floor (long double __x)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`
`floor (_Tp __x)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & flush (basic_ostream< _CharT, _Traits >`
`& __os)`
- `float fmod (float __x, float __y)`
- `long double fmod (long double __x, long double __y)`
- `template<typename _Iter, typename _Funct >`
`_Funct for_each (_Iter, _Iter, _Funct)`
- `template<typename _InputIterator, typename _Function >`
`_Function for_each (_InputIterator __first, _InputIterator __last, _Function __f)`
- `template<typename _Tp >`
`_Tp && forward (typename std::remove_reference< _Tp >::__type & __t)`

- `template<typename _Tp >`
`_Tp && forward (typename std::remove_reference< _Tp >::type &&__t)`
- `template<typename... _Elements>`
`tuple< _Elements &&...> forward_as_tuple (_Elements &&...__args)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type frexp (_Tp __x, int *__exp)`
- `long double frexp (long double __x, int *__exp)`
- `float frexp (float __x, int *__exp)`
- `template<typename _Container >`
`front_insert_iterator< _Container > front_inserter (_Container &__x)`
- `const error_category & future_category ()`
- `template<typename _Filter, typename _Generator >`
`void generate (_Filter, _Filter, _Generator)`
- `template<typename _ForwardIterator, typename _Generator >`
`void generate (_ForwardIterator __first, _ForwardIterator __last, _Generator __gen)`
- `template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator >`
`_RealType generate_canonical (_UniformRandomNumberGenerator &__g)`
- `template<typename _OIter, typename _Size, typename _Generator >`
`_OIter generate_n (_OIter, _Size, _Generator)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`
`_OutputIterator generate_n (_OutputIterator __first, _Size __n, _Generator __gen)`
- `const error_category & generic_category () throw ()`
- `template<std::size_t __i, typename... _Elements>`
`__add_ref< typename tuple_element< __i, tuple< _Elements...> >::type >::type get (tuple< _Elements...> &__t)`
- `template<std::size_t __i, typename... _Elements>`
`__add_c_ref< typename tuple_element< __i, tuple< _Elements...> >::type >::type get (const tuple< _Elements...> &__t)`
- `template<std::size_t _Int, class _Tp1, class _Tp2 >`
`tuple_element< _Int, std::pair< _Tp1, _Tp2 > >::type & get (std::pair< _Tp1, _Tp2 > &__in)`
- `template<std::size_t _Int, class _Tp1, class _Tp2 >`
`const tuple_element< _Int, std::pair< _Tp1, _Tp2 > >::type & get (const std::pair< _Tp1, _Tp2 > &__in)`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>`
`_Tp & get (array< _Tp, _Nm > &__arr)`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>`
`const _Tp & get (const array< _Tp, _Nm > &__arr)`
- `template<typename _Del, typename _Tp, _Lock_policy _Lp>`
`_Del * get_deleter (const __shared_ptr< _Tp, _Lp > &__p)`
- `template<typename _MoneyT >`
`_Get_money< _MoneyT > get_money (_MoneyT &__mon, bool __intl=false)`

- `template<typename _Tp >`
`pair< _Tp *, ptrdiff_t > get_temporary_buffer (ptrdiff_t __len)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str)`
- `template<>`
`basic_istream< char > & getline (basic_istream< char > &__in, basic_string< char > &__str, char __delim)`
- `template<>`
`basic_istream< wchar_t > & getline (basic_istream< wchar_t > &__in, basic_string< wchar_t > &__str, wchar_t __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str)`
- `template<typename _Facet >`
`bool has_facet (const locale &__loc) throw ()`
- `ios_base & hex (ios_base &__base)`
- `template<typename _Tp >`
`constexpr _Tp imag (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type imag (_Tp)`
- `template<typename _InputIterator1, typename _InputIterator2 >`
`bool includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare >`
`bool includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2 >`
`bool includes (_Iter1, _Iter1, _Iter2, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _Compare >`
`bool includes (_Iter1, _Iter1, _Iter2, _Iter2, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _BinaryOperation1, typename _BinaryOperation2 >`
`_Tp inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init, _BinaryOperation1 __binary_op1, _BinaryOperation2 __binary_op2)`

- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp >`
`_Tp inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _-`
`InputIterator2 __first2, _Tp __init)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`void inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __-`
`middle, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _BidirectionalIterator >`
`void inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __-`
`middle, _BidirectionalIterator __last)`
- `template<typename _BIter >`
`void inplace_merge (_BIter, _BIter, _BIter)`
- `template<typename _BIter, typename _Compare >`
`void inplace_merge (_BIter, _BIter, _BIter, _Compare)`
- `template<typename _Container, typename _Iterator >`
`insert_iterator< _Container > insert (_Container & __x, _Iterator __i)`
- `ios_base & internal (ios_base & __base)`
- `template<typename _ForwardIterator, typename _Tp >`
`void iota (_ForwardIterator __first, _ForwardIterator __last, _Tp __value)`
- `template<typename _RandomAccessIterator >`
`bool is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`bool is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _-`
`Compare __comp)`
- `template<typename _RAIter >`
`bool is_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`bool is_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator >`
`_RandomAccessIterator is_heap_until (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`_RandomAccessIterator is_heap_until (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RAIter >`
`_RAIter is_heap_until (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`_RAIter is_heap_until (_RAIter, _RAIter, _Compare)`
- `template<typename _Iter, typename _Predicate >`
`bool is_partitioned (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`
`bool is_partitioned (_InputIterator __first, _InputIterator __last, _Predicate __-`
`pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`bool is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _-`
`ForwardIterator2 __first2)`

- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>`
`bool is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _-`
`ForwardIterator2 __first2, _BinaryPredicate __pred)`
- `template<typename _FIter1, typename _FIter2>`
`bool is_permutation (_FIter1, _FIter1, _FIter2)`
- `template<typename _FIter1, typename _FIter2, typename _BinaryPredicate>`
`bool is_permutation (_FIter1, _FIter1, _FIter2, _BinaryPredicate)`
- `template<typename _ForwardIterator>`
`bool is_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare>`
`bool is_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __-`
`comp)`
- `template<typename _FIter>`
`bool is_sorted (_FIter, _FIter)`
- `template<typename _FIter, typename _Compare>`
`bool is_sorted (_FIter, _FIter, _Compare)`
- `template<typename _ForwardIterator, typename _Compare>`
`_ForwardIterator is_sorted_until (_ForwardIterator __first, _ForwardIterator __-`
`last, _Compare __comp)`
- `template<typename _ForwardIterator>`
`_ForwardIterator is_sorted_until (_ForwardIterator __first, _ForwardIterator __-`
`last)`
- `template<typename _FIter>`
`_FIter is_sorted_until (_FIter, _FIter)`
- `template<typename _FIter, typename _Compare>`
`_FIter is_sorted_until (_FIter, _FIter, _Compare)`
- `template<typename _CharT>`
`bool isalnum (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`
`bool isalpha (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`
`bool iscntrl (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`
`bool isdigit (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`
`bool isgraph (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`
`bool islower (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`
`bool isprint (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`
`bool ispunct (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`
`bool isspace (_CharT __c, const locale &__loc)`

- `template<typename _CharT >`
`bool isupper (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`
`bool isxdigit (_CharT __c, const locale &__loc)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`void iter_swap (_ForwardIterator1 __a, _ForwardIterator2 __b)`
- `template<typename _FIter1, typename _FIter2 >`
`void iter_swap (_FIter1, _FIter2)`
- `template<typename _Tp >`
`_Tp kill_dependency (_Tp __y)`
- `long double ldexp (long double __x, int __exp)`
- `float ldexp (float __x, int __exp)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type ldexp (_Tp __x, int __exp)`
- `ios_base & left (ios_base &__base)`
- `template<typename _Iter1, typename _Iter2 >`
`bool lexicographical_compare (_Iter1, _Iter1, _Iter2, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _Compare >`
`bool lexicographical_compare (_Iter1, _Iter1, _Iter2, _Iter2, _Compare)`
- `template<typename _II1, typename _II2, typename _Compare >`
`bool lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _Compare __comp)`
- `template<typename _II1, typename _II2 >`
`bool lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)`
- `template<typename _L1, typename _L2, typename... _L3>`
`void lock (_L1 &__l1, _L2 &__l2, _L3 &...__l3)`
- `template<typename _Tp >`
`complex< _Tp > log (const complex< _Tp > &)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type log (_Tp __x)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Log, _ValArray, _Tp >, _Tp > log (const valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Log, _Expr, _Dom >, typename _Dom::value_type > log (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `float log (float __x)`
- `long double log (long double __x)`
- `template<typename _Tp >`
`complex< _Tp > log10 (const complex< _Tp > &)`

- `template<typename _Tp >`
`_Expr< _UnClos< _Log10, _ValArray, _Tp >, _Tp > log10 (const valarray<`
`_Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Log10, _Expr, _Dom >, typename _Dom::value_type >`
`log10 (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `float log10 (float __x)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`
`log10 (_Tp __x)`
- `long double log10 (long double __x)`
- `template<typename _ForwardIterator, typename _Tp >`
`_ForwardIterator lower_bound (_ForwardIterator __first, _ForwardIterator __-`
`last, const _Tp &__val)`
- `template<typename _FIter, typename _Tp >`
`_FIter lower_bound (_FIter, _FIter, const _Tp &)`
- `template<typename _FIter, typename _Tp, typename _Compare >`
`_FIter lower_bound (_FIter, _FIter, const _Tp &, _Compare)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`_ForwardIterator lower_bound (_ForwardIterator __first, _ForwardIterator __-`
`last, const _Tp &__val, _Compare __comp)`
- `error_code make_error_code (future_errc __errc)`
- `error_code make_error_code (errc __e)`
- `error_condition make_error_condition (future_errc __errc)`
- `error_condition make_error_condition (errc __e)`
- `template<typename _Ex >`
`exception_ptr make_exception_ptr (_Ex __ex) throw ()`
- `template<typename _RandomAccessIterator >`
`void make_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`
`last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void make_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`
`last, _Compare __comp)`
- `template<typename _RAIter >`
`void make_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`void make_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _Iterator >`
`move_iterator< _Iterator > make_move_iterator (const _Iterator &__i)`
- `template<class _T1, class _T2 >`
`pair< typename __decay_and_strip< _T1 >::__type, typename __decay_and_-`
`strip< _T2 >::__type > make_pair (_T1 &&__x, _T2 &&__y)`
- `template<typename _Tp, typename... _Args>`
`shared_ptr< _Tp > make_shared (_Args &&...__args)`

- `template<typename... _Elements>`
`tuple< typename __decay_and_strip< _Elements >::__type...> make_tuple`
`(_Elements &&...__args)`
- `template<typename _Tp >`
`const _Tp & max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`
`const _Tp & max (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`
`_Tp max (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`
`_Tp max (initializer_list< _Tp >, _Compare)`
- `template<typename _FIter >`
`_FIter max_element (_FIter, _FIter)`
- `template<typename _FIter, typename _Compare >`
`_FIter max_element (_FIter, _FIter, _Compare)`
- `template<typename _ForwardIterator >`
`_ForwardIterator max_element (_ForwardIterator __first, _ForwardIterator __-`
`last)`
- `template<typename _ForwardIterator, typename _Compare >`
`_ForwardIterator max_element (_ForwardIterator __first, _ForwardIterator __-`
`last, _Compare __comp)`
- `template<typename _Tp, typename _Class >`
`_Mem_fn< _Tp _Class::* > mem_fn (_Tp _Class::*__pm)`
- `template<typename _Ret, typename _Tp >`
`mem_fun_t< _Ret, _Tp > mem_fun (_Ret(_Tp::*__f)())`
- `template<typename _Ret, typename _Tp, typename _Arg >`
`mem_fun1_t< _Ret, _Tp, _Arg > mem_fun (_Ret(_Tp::*__f)(_Arg))`
- `template<typename _Ret, typename _Tp >`
`mem_fun_ref_t< _Ret, _Tp > mem_fun_ref (_Ret(_Tp::*__f)())`
- `template<typename _Ret, typename _Tp, typename _Arg >`
`mem_fun1_ref_t< _Ret, _Tp, _Arg > mem_fun_ref (_Ret(_Tp::*__f)(_Arg))`
- `void * memchr (void *__s, int __c, size_t __n)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator merge (_InputIterator1 __first1, _InputIterator1 __last1, _-`
`InputIterator2 __first2, InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`
`name _Compare >`
`_OutputIterator merge (_InputIterator1 __first1, _InputIterator1 __last1, _-`
`InputIterator2 __first2, InputIterator2 __last2, _OutputIterator __result, _-`
`Compare __comp)`

- `template<typename _Tp >`
`const _Tp & min (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`
`const _Tp & min (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`
`_Tp min (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`
`_Tp min (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter >`
`_Filter min_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`
`_Filter min_element (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator >`
`_ForwardIterator min_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`_ForwardIterator min_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Tp >`
`pair< const _Tp &, const _Tp & > minmax (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`
`pair< const _Tp &, const _Tp & > minmax (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`
`pair< _Tp, _Tp > minmax (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`
`pair< _Tp, _Tp > minmax (initializer_list< _Tp >, _Compare)`
- `template<typename _ForwardIterator, typename _Compare >`
`pair< _ForwardIterator, _ForwardIterator > minmax_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator >`
`pair< _ForwardIterator, _ForwardIterator > minmax_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _Filter >`
`pair< _Filter, _Filter > minmax_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`
`pair< _Filter, _Filter > minmax_element (_Filter, _Filter, _Compare)`
- `template<typename _Iter1, typename _Iter2 >`
`pair< _Iter1, _Iter2 > mismatch (_Iter1, _Iter1, _Iter2)`
- `template<typename _InputIterator1, typename _InputIterator2 >`
`pair< _InputIterator1, _InputIterator2 > mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`
`pair< _InputIterator1, _InputIterator2 > mismatch (_InputIterator1 __first1,`

- `_InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`
`pair< _Iter1, _Iter2 > mismatch (_Iter1, _Iter1, _Iter2, _BinaryPredicate)`
- `float modf (float __x, float * __iptr)`
- `long double modf (long double __x, long double * __iptr)`
- `template<typename _II, typename _OI >`
`_OI move (_II __first, _II __last, _OI __result)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * > move (_Deque_iterator< _Tp, _Tp &, _Tp * > __first, _Deque_iterator< _Tp, _Tp &, _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * > move (_Deque_iterator< _Tp, const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const _Tp &, const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`
`std::remove_reference< _Tp >::type && move (_Tp && __t)`
- `template<typename _BI1, typename _BI2 >`
`_BI2 move_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * > move_backward (_Deque_iterator< _Tp, _Tp &, _Tp * > __first, _Deque_iterator< _Tp, _Tp &, _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * > move_backward (_Deque_iterator< _Tp, const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const _Tp &, const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _ForwardIterator >`
`_ForwardIterator next (_ForwardIterator __x, typename iterator_traits< _ForwardIterator >::difference_type __n=1)`
- `template<typename _BidirectionalIterator >`
`bool next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`bool next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _BIter >`
`bool next_permutation (_BIter, _BIter)`
- `template<typename _BIter, typename _Compare >`
`bool next_permutation (_BIter, _BIter, _Compare)`
- `ios_base & noboolalpha (ios_base & __base)`
- `template<typename _InputIterator, typename _Predicate >`
`bool none_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`

- `template<typename _Iter, typename _Predicate >`
`bool none_of (_Iter, _Iter, _Predicate)`
- `template<typename _Tp >`
`_Tp norm (const complex< _Tp > &)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type norm (_Tp __x)`
- `ios_base & noshowbase (ios_base &__base)`
- `ios_base & noshowpoint (ios_base &__base)`
- `ios_base & noshowpos (ios_base &__base)`
- `ios_base & noskipws (ios_base &__base)`
- `template<typename _Predicate >`
`unary_negate< _Predicate > not1 (const _Predicate &__pred)`
- `template<typename _Predicate >`
`binary_negate< _Predicate > not2 (const _Predicate &__pred)`
- `ios_base & nunitbuf (ios_base &__base)`
- `ios_base & nouppercase (ios_base &__base)`
- `template<typename _RAIter >`
`void nth_element (_RAIter, _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`void nth_element (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator >`
`void nth_element (_RandomAccessIterator __first, _RandomAccessIterator __-`
`nth, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void nth_element (_RandomAccessIterator __first, _RandomAccessIterator __-`
`nth, _RandomAccessIterator __last, _Compare __comp)`
- `ios_base & oct (ios_base &__base)`
- `bool operator!= (const error_code &__lhs, const error_code &__rhs)`
- `template<typename _RealType >`
`bool operator!= (const std::weibull_distribution< _RealType > &__d1, const`
`std::weibull_distribution< _RealType > &__d2)`
- `bool operator!= (const error_code &__lhs, const error_condition &__rhs)`
- `bool operator!= (const error_condition &__lhs, const error_code &__rhs)`
- `bool operator!= (const error_condition &__lhs, const error_condition &__rhs)`
- `bool operator!= (thread::id __x, thread::id __y)`
- `template<typename... _TElements, typename... _UElements>`
`bool operator!= (const tuple< _TElements...> &__t, const tuple< _-`
`UElements...> &__u)`
- `template<typename _RealType >`
`bool operator!= (const std::extreme_value_distribution< _RealType > &__d1,`
`const std::extreme_value_distribution< _RealType > &__d2)`
- `template<typename _T1, typename _T2 >`
`bool operator!= (const allocator< _T1 > &, const allocator< _T2 > &)`

- `template<typename _Tp >`
`bool operator!= (const allocator< _Tp > &, const allocator< _Tp > &)`
- `bool operator!= (const std::bernoulli_distribution &__d1, const std::bernoulli_distribution &__d2)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`bool operator!= (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`
`bool operator!= (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator!= (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator!= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _RealType >`
`bool operator!= (const std::piecewise_constant_distribution< _RealType > &__d1, const std::piecewise_constant_distribution< _RealType > &__d2)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator!= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _Iterator >`
`bool operator!= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator!= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator!= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`bool operator!= (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool operator!= (nullptr_t, const unique_ptr< _Tp, _Dp > &__y)`
- `template<typename _Val >`
`bool operator!= (const _List_iterator< _Val > &__x, const _List_const_iterator< _Val > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator!= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`

- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool operator!= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const`
`map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool operator!= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const`
`multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator!= (const multiset< _Key, _Compare, _Alloc > &__x, const mul-`
`tiset< _Key, _Compare, _Alloc > &__y)`
- `template<class _T1, class _T2 >`
`constexpr bool operator!= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2`
`> &__y)`
- `template<typename _Tp >`
`bool operator!= (const _Fwd_list_iterator< _Tp > &__x, const _Fwd_list_-`
`const_iterator< _Tp > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool operator!= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq >`
`&__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator!= (const forward_list< _Tp, _Alloc > &__lx, const forward_list<`
`_Tp, _Alloc > &__ly)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator!= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key,`
`_Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool operator!= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq >`
`&__y)`
- `template<typename _Tp >`
`_Expr< _BinClos< __not_equal_to, _ValArray, _ValArray, _Tp, _Tp >, type-`
`name __fun< __not_equal_to, _Tp >::result_type > operator!= (const valar-`
`ray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __not_equal_to, _ValArray, _Constant, _Tp, _Tp >, type-`
`name __fun< __not_equal_to, _Tp >::result_type > operator!= (const valar-`
`ray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Res, typename... _Args>`
`bool operator!= (const function< _Res(_Args...)> &__f, nullptr_t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __not_equal_to, _Constant, _ValArray, _Tp, _Tp >, type-`
`name __fun< __not_equal_to, _Tp >::result_type > operator!= (const _Tp &_-`
`_t, const valarray< _Tp > &__v)`
- `template<typename _Val >`
`bool operator!= (const _Rb_tree_iterator< _Val > &__x, const _Rb_tree_-`
`const_iterator< _Val > &__y)`

- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >`
`bool operator!= (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _RealType >`
`bool operator!= (const std::lognormal_distribution< _RealType > &__d1, const std::lognormal_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool operator!= (const std::gamma_distribution< _RealType > &__d1, const std::gamma_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool operator!= (const std::chi_squared_distribution< _RealType > &__d1, const std::chi_squared_distribution< _RealType > &__d2)`
- `template<typename _IntType >`
`bool operator!= (const std::uniform_real_distribution< _IntType > &__d1, const std::uniform_real_distribution< _IntType > &__d2)`
- `template<typename _RealType >`
`bool operator!= (const std::normal_distribution< _RealType > &__d1, const std::normal_distribution< _RealType > &__d2)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>`
`bool operator!= (const std::linear_congruential_engine< _UIntType, __a, __c, __m > &__lhs, const std::linear_congruential_engine< _UIntType, __a, __c, __m > &__rhs)`
- `template<typename _StateT >`
`bool operator!= (const fpos< _StateT > &__lhs, const fpos< _StateT > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r>`
`bool operator!= (const std::discard_block_engine< _RandomNumberEngine, __p, __r > &__lhs, const std::discard_block_engine< _RandomNumberEngine, __p, __r > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType >`
`bool operator!= (const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__lhs, const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator!= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<class _Tp, class _CharT, class _Traits, class _Dist >`
`bool operator!= (const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__x, const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__y)`
- `template<typename _CharT, typename _Traits >`
`bool operator!= (const istreambuf_iterator< _CharT, _Traits > &__a, const istreambuf_iterator< _CharT, _Traits > &__b)`

- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool operator!= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool operator!= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`bool operator!= (const __unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__x, const __unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`bool operator!= (const __unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__x, const __unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>`
`bool operator!= (const std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__lhs, const std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__rhs)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`
`bool operator!= (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`bool operator!= (const __unordered_set< _Value, _Hash, _Pred, _Alloc, __cache_hash_code > &__x, const __unordered_set< _Value, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`bool operator!= (const __unordered_multiset< _Value, _Hash, _Pred, _Alloc, __cache_hash_code > &__x, const __unordered_multiset< _Value, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`
`bool operator!= (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Dom >`
`_Expr< _BinClos< __not_equal_to, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __not_equal_to, typename _Dom::value_type >::result_type > operator!= (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __not_equal_to, _Expr, _ValArray, _Dom, typename`

- ```

_Dom::value_type >, typename __fun< __not_equal_to, typename _-
Dom::value_type >::result_type > operator!= (const _Expr< _Dom, typename
_Dom::value_type > &__e, const valarray< typename _Dom::value_type >
&__v)

```
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator!= (const _CharT *__lhs, const basic\_string< _CharT, _Traits, _-`  
`_Alloc > &__rhs)`
  - `template<typename _IntType >`  
`bool operator!= (const std::poisson\_distribution< _IntType > &__d1, const`  
`std::poisson\_distribution< _IntType > &__d2)`
  - `template<typename _UIntType, size_t __w, size_t __s, size_t __r>`  
`bool operator!= (const std::subtract\_with\_carry\_engine< _UIntType, __w, _-`  
`s, __r > &__lhs, const std::subtract\_with\_carry\_engine< _UIntType, __w, __s,`  
`__r > &__rhs)`
  - `template<typename _IntType >`  
`bool operator!= (const std::uniform\_int\_distribution< _IntType > &__d1, const`  
`std::uniform\_int\_distribution< _IntType > &__d2)`
  - `template<class _Dom >`  
`_Expr< _BinClos< __not_equal_to, _Constant, _Expr, typename _-`  
`Dom::value_type, _Dom >, typename __fun< __not_equal_to, type-`  
`name _Dom::value_type >::result_type > operator!= (const typename`  
`_Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type >`  
`&__v)`
  - `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __not_equal_to, _Expr, _Expr, _Dom1, _Dom2 >, type-`  
`name __fun< __not_equal_to, typename _Dom1::value_type >::result_type >`  
`operator!= (const _Expr< _Dom1, typename _Dom1::value_type > &__v,`  
`const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
  - `template<typename _RealType >`  
`bool operator!= (const std::exponential\_distribution< _RealType > &__d1,`  
`const std::exponential\_distribution< _RealType > &__d2)`
  - `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`  
`bool operator!= (const unordered\_map< _Key, _Tp, _Hash, _Pred, _Alloc >`  
`&__x, const unordered\_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
  - `template<typename _RealType >`  
`bool operator!= (const std::piecewise\_linear\_distribution< _RealType > &__-`  
`d1, const std::piecewise\_linear\_distribution< _RealType > &__d2)`
  - `template<class _Dom >`  
`_Expr< _BinClos< __not_equal_to, _ValArray, _Expr, typename _-`  
`Dom::value_type, _Dom >, typename __fun< __not_equal_to, typename`  
`_Dom::value_type >::result_type > operator!= (const valarray< typename`  
`_Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type`  
`> &__e)`
  - `template<typename _RealType >`  
`bool operator!= (const std::cauchy\_distribution< _RealType > &__d1, const`  
`std::cauchy\_distribution< _RealType > &__d2)`

- `template<typename _BiIter >`  
`bool operator!= (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool operator!= (const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool operator!= (const sub_match< _Bi_iter > &__lhs, const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`  
`bool operator!= (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool operator!= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter >`  
`bool operator!= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool operator!= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`  
`bool operator!= (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Bi_iter, class _Allocator >`  
`bool operator!= (const match_results< _Bi_iter, _Allocator > &__m1, const match_results< _Bi_iter, _Allocator > &__m2)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator!= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b)`
- `template<typename _Tp >`  
`bool operator!= (const shared_ptr< _Tp > &__a, nullptr_t)`
- `template<typename _Tp >`  
`bool operator!= (nullptr_t, const shared_ptr< _Tp > &__b)`
- `template<typename _Res, typename... _Args>`  
`bool operator!= (nullptr_t, const function< _Res(_Args...)> &__f)`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`  
`bool operator!= (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool operator!= (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool operator!= (nullptr_t, const __shared_ptr< _Tp, _Lp > &__b)`



- `template<typename _RealType >`  
`bool operator!= (const std::fisher_f_distribution< _RealType > &__d1, const`  
`std::fisher_f_distribution< _RealType > &__d2)`
- `template<typename _IntType >`  
`bool operator!= (const std::discrete_distribution< _IntType > &__d1, const`  
`std::discrete_distribution< _IntType > &__d2)`
- `template<typename _IntType >`  
`bool operator!= (const std::binomial_distribution< _IntType > &__d1, const`  
`std::binomial_distribution< _IntType > &__d2)`
- `template<typename _RandomNumberEngine, size_t __k>`  
`bool operator!= (const std::shuffle_order_engine< _RandomNumberEngine, _`  
`__k > &__lhs, const std::shuffle_order_engine< _RandomNumberEngine, _`  
`__k > &__rhs)`
- `template<typename _Tp, std::size_t _Nm>`  
`bool operator!= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm`  
`> &__two)`
- `template<typename _IntType >`  
`bool operator!= (const std::geometric_distribution< _IntType > &__d1, const`  
`std::geometric_distribution< _IntType > &__d2)`
- `template<typename _IntType >`  
`bool operator!= (const std::negative_binomial_distribution< _IntType > &__d1,`  
`const std::negative_binomial_distribution< _IntType > &__d2)`
- `template<typename _RealType >`  
`bool operator!= (const std::student_t_distribution< _RealType > &__d1, const`  
`std::student_t_distribution< _RealType > &__d2)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __modulus, _ValArray, _ValArray, _Tp, _Tp >, typename`  
`__fun< __modulus, _Tp >::result_type > operator% (const valarray< _Tp >`  
`&__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __modulus, _ValArray, _Constant, _Tp, _Tp >, typename`  
`__fun< __modulus, _Tp >::result_type > operator% (const valarray< _Tp >`  
`&__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __modulus, _Constant, _ValArray, _Tp, _Tp >, typename`  
`__fun< __modulus, _Tp >::result_type > operator% (const _Tp &__t, const`  
`valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __modulus, _ValArray, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __modulus, typename _Dom::value_type`  
`>::result_type > operator% (const valarray< typename _Dom::value_type >`  
`&__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __modulus, _Expr, _Expr, _Dom1, _Dom2 >, typename _`  
`__fun< __modulus, typename _Dom1::value_type >::result_type > operator%`

```
(const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr<
_Dom2, typename _Dom2::value_type > &__w)
```

- `template<class _Dom >`  
`_Expr< _BinClos< __modulus, _Expr, _ValArray, _Dom, typename _-`  
`Dom::value_type >, typename __fun< __modulus, typename _Dom::value_-`  
`type >::result_type > operator% (const _Expr< _Dom, typename _-`  
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__-`  
`_v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __modulus, _Expr, _Constant, _Dom, typename _-`  
`Dom::value_type >, typename __fun< __modulus, typename _Dom::value_-`  
`type >::result_type > operator% (const _Expr< _Dom, typename _-`  
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __modulus, _Constant, _Expr, typename _Dom::value_-`  
`type, _Dom >, typename __fun< __modulus, typename _Dom::value_type`  
`>::result_type > operator% (const typename _Dom::value_type &__t, const`  
`_Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_and, _ValArray, _ValArray, _Tp, _Tp >, type-`  
`name __fun< __bitwise_and, _Tp >::result_type > operator& (const valar-`  
`ray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_and, _ValArray, _Constant, _Tp, _Tp >, type-`  
`name __fun< __bitwise_and, _Tp >::result_type > operator& (const valar-`  
`ray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_and, _Constant, _ValArray, _Tp, _Tp >, type-`  
`name __fun< __bitwise_and, _Tp >::result_type > operator& (const _Tp &__-`  
`_t, const valarray< _Tp > &__v)`
- `constexpr _Ios_Fmtflags operator& (_Ios_Fmtflags __a, _Ios_Fmtflags __b)`
- `constexpr _Ios_Openmode operator& (_Ios_Openmode __a, _Ios_Openmode`  
`__b)`
- `constexpr _Ios_Iostate operator& (_Ios_Iostate __a, _Ios_Iostate __b)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __bitwise_and, _Expr, _Expr, _Dom1, _Dom2 >, typename`  
`__fun< __bitwise_and, typename _Dom1::value_type >::result_type > opera-`  
`tor& (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _-`  
`Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_and, _Constant, _Expr, typename _Dom::value_-`  
`type, _Dom >, typename __fun< __bitwise_and, typename _Dom::value_type`  
`>::result_type > operator& (const typename _Dom::value_type &__t, const`  
`_Expr< _Dom, typename _Dom::value_type > &__v)`

- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_and, _Expr, _ValArray, _Dom, typename`  
`_Dom::value_type >, typename __fun< __bitwise_and, typename _-`  
`Dom::value_type >::result_type > operator& (const _Expr< _Dom, typename`  
`_Dom::value_type > &__e, const valarray< typename _Dom::value_type >`  
`&__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_and, _ValArray, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __bitwise_and, typename _Dom::value_type`  
`>::result_type > operator& (const valarray< typename _Dom::value_type >`  
`&__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_and, _Expr, _Constant, _Dom, typename`  
`_Dom::value_type >, typename __fun< __bitwise_and, typename _-`  
`Dom::value_type >::result_type > operator& (const _Expr< _Dom, typename`  
`_Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_and, _ValArray, _ValArray, _Tp, _Tp >, type-`  
`name __fun< __logical_and, _Tp >::result_type > operator&& (const valar-`  
`ray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_and, _ValArray, _Constant, _Tp, _Tp >, type-`  
`name __fun< __logical_and, _Tp >::result_type > operator&& (const valar-`  
`ray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_and, _Constant, _ValArray, _Tp, _Tp >, type-`  
`name __fun< __logical_and, _Tp >::result_type > operator&& (const _Tp`  
`&__t, const valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_and, _Expr, _ValArray, _Dom, typename _-`  
`Dom::value_type >, typename __fun< __logical_and, typename _Dom::value_`  
`type >::result_type > operator&& (const _Expr< _Dom, typename _-`  
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__`  
`_v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_and, _Constant, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __logical_and, typename _Dom::value_type`  
`>::result_type > operator&& (const typename _Dom::value_type &__t, const`  
`_Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_and, _ValArray, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __logical_and, typename _Dom::value_type`  
`>::result_type > operator&& (const valarray< typename _Dom::value_type >`  
`&__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`

- `template<class _Dom >`  
`_Expr< _BinClos< __logical_and, _Expr, _Constant, _Dom, typename _`  
`Dom::value_type >, typename __fun< __logical_and, typename _Dom::value_`  
`type >::result_type > operator&& (const _Expr< _Dom, typename _`  
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __logical_and, _Expr, _Expr, _Dom1, _Dom2 >, typename`  
`__fun< __logical_and, typename _Dom1::value_type >::result_type > opera-`  
`tor&& (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const`  
`_Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `const _Ios_Fmtflags & operator&= (_Ios_Fmtflags &__a, _Ios_Fmtflags __b)`
- `const _Ios_Openmode & operator&= (_Ios_Openmode &__a, _Ios_Openmode`  
`__b)`
- `const _Ios_Iostate & operator&= (_Ios_Iostate &__a, _Ios_Iostate __b)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __multiplies, _ValArray, _ValArray, _Tp, _Tp >, typename`  
`__fun< __multiplies, _Tp >::result_type > operator* (const valarray< _Tp >`  
`&__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __multiplies, _ValArray, _Constant, _Tp, _Tp >, typename`  
`__fun< __multiplies, _Tp >::result_type > operator* (const valarray< _Tp >`  
`&__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __multiplies, _Constant, _ValArray, _Tp, _Tp >, typename`  
`__fun< __multiplies, _Tp >::result_type > operator* (const _Tp &__t, const`  
`valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __multiplies, _ValArray, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __multiplies, typename _Dom::value_type`  
`>::result_type > operator* (const valarray< typename _Dom::value_type >`  
`&__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __multiplies, _Expr, _Expr, _Dom1, _Dom2 >, type-`  
`name __fun< __multiplies, typename _Dom1::value_type >::result_type >`  
`operator* (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const`  
`_Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __multiplies, _Expr, _Constant, _Dom, typename _`  
`Dom::value_type >, typename __fun< __multiplies, typename _Dom::value_`  
`type >::result_type > operator* (const _Expr< _Dom, typename _`  
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __multiplies, _Constant, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __multiplies, typename _Dom::value_type`

>::result\_type > **operator\*** (const typename \_Dom::value\_type &\_\_t, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_v)

- template<class \_Dom >  
\_Expr< \_BinClos< \_\_multiplies, \_Expr, \_ValArray, \_Dom, typename \_Dom::value\_type >, typename \_\_fun< \_\_multiplies, typename \_Dom::value\_type >::result\_type > **operator\*** (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e, const [valarray](#)< typename \_Dom::value\_type > &\_\_v)
- template<typename \_CharT, typename \_Traits, typename \_Alloc >  
[basic\\_string](#)< \_CharT, \_Traits, \_Alloc > **operator+** (const \_CharT \*\_\_lhs, [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &&\_\_rhs)
- \_Bit\_iterator **operator+** (ptrdiff\_t \_\_n, const \_Bit\_iterator &\_\_x)
- template<typename \_CharT, typename \_Traits, typename \_Alloc >  
[basic\\_string](#)< \_CharT, \_Traits, \_Alloc > **operator+** (const [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_lhs, const [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_rhs)
- template<typename \_CharT, typename \_Traits, typename \_Alloc >  
[basic\\_string](#)< \_CharT, \_Traits, \_Alloc > **operator+** (const \_CharT \*\_\_lhs, const [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_rhs)
- template<typename \_Tp >  
[complex](#)< \_Tp > **operator+** (const [complex](#)< \_Tp > &\_\_x)
- template<typename \_CharT, typename \_Traits, typename \_Alloc >  
[basic\\_string](#)< \_CharT, \_Traits, \_Alloc > **operator+** (\_CharT \_\_lhs, const [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_rhs)
- template<typename \_CharT, typename \_Traits, typename \_Alloc >  
[basic\\_string](#)< \_CharT, \_Traits, \_Alloc > **operator+** (const [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_lhs, const \_CharT \*\_\_rhs)
- template<typename \_CharT, typename \_Traits, typename \_Alloc >  
[basic\\_string](#)< \_CharT, \_Traits, \_Alloc > **operator+** (const [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_lhs, \_CharT \_\_rhs)
- template<typename \_CharT, typename \_Traits, typename \_Alloc >  
[basic\\_string](#)< \_CharT, \_Traits, \_Alloc > **operator+** ([basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &&\_\_lhs, const [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_rhs)
- template<typename \_CharT, typename \_Traits, typename \_Alloc >  
[basic\\_string](#)< \_CharT, \_Traits, \_Alloc > **operator+** (const [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_lhs, [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &&\_\_rhs)
- template<typename \_Tp, typename \_Ref, typename \_Ptr >  
[Deque\\_iterator](#)< \_Tp, \_Ref, \_Ptr > **operator+** (ptrdiff\_t \_\_n, const [Deque\\_iterator](#)< \_Tp, \_Ref, \_Ptr > &\_\_x)
- template<typename \_CharT, typename \_Traits, typename \_Alloc >  
[basic\\_string](#)< \_CharT, \_Traits, \_Alloc > **operator+** ([basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &&\_\_lhs, [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &&\_\_rhs)

- `template<typename _Tp >`  
`_Expr< _BinClos< __plus, _ValArray, _Constant, _Tp, _Tp >, typename __-`  
`fun< __plus, _Tp >::result_type > operator+ (const valarray< _Tp > &__v,`  
`const _Tp &__t)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic\_string< _CharT, _Traits, _Alloc > operator+ (_CharT __lhs, basic\_-`  
`string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __plus, _Constant, _ValArray, _Tp, _Tp >, typename __-`  
`fun< __plus, _Tp >::result_type > operator+ (const _Tp &__t, const valarray<`  
`_Tp > &__v)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic\_string< _CharT, _Traits, _Alloc > operator+ (basic\_string< _CharT, _-`  
`_Traits, _Alloc > &&__lhs, _CharT __rhs)`
- `template<typename _Iterator >`  
`reverse\_iterator< _Iterator > operator+ (typename reverse\_iterator< _Iterator`  
`>::difference_type __n, const reverse\_iterator< _Iterator > &__x)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __plus, _ValArray, _ValArray, _Tp, _Tp >, typename __-`  
`fun< __plus, _Tp >::result_type > operator+ (const valarray< _Tp > &__v,`  
`const valarray< _Tp > &__w)`
- `template<typename _Iterator >`  
`move\_iterator< _Iterator > operator+ (typename move\_iterator< _Iterator`  
`>::difference_type __n, const move\_iterator< _Iterator > &__x)`
- `template<class _Dom >`  
`_Expr< _BinClos< __plus, _Constant, _Expr, typename _Dom::value_type, _-`  
`Dom >, typename __fun< __plus, typename _Dom::value_type >::result_type`  
`> operator+ (const typename _Dom::value_type &__t, const _Expr< _Dom,`  
`typename _Dom::value_type > &__v)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic\_string< _CharT, _Traits, _Alloc > operator+ (basic\_string< _CharT, _-`  
`_Traits, _Alloc > &&__lhs, const _CharT *__rhs)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __plus, _Expr, _Expr, _Dom1, _Dom2 >, typename __-`  
`fun< __plus, typename _Dom1::value_type >::result_type > operator+ (const`  
`_Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2,`  
`typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __plus, _ValArray, _Expr, typename _Dom::value_type, _-`  
`Dom >, typename __fun< __plus, typename _Dom::value_type >::result_type`  
`> operator+ (const valarray< typename _Dom::value_type > &__v, const _-`  
`Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< __plus, _Expr, _Constant, _Dom, typename _Dom::value _-`  
`type >, typename __fun< __plus, typename _Dom::value_type >::result_type`

> **operator+** (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_v, const typename \_Dom::value\_type &\_\_t)

- template<class \_Dom >  
\_Expr< \_BinClos< \_\_plus, \_Expr, \_ValArray, \_Dom, typename \_Dom::value\_type >, typename \_\_fun< \_\_plus, typename \_Dom::value\_type >::result\_type > **operator+** (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e, const [valarray](#)< typename \_Dom::value\_type > &\_\_v)
- \_Bit\_const\_iterator **operator+** (ptrdiff\_t \_\_n, const \_Bit\_const\_iterator &\_\_x)
- template<typename \_Iterator >  
auto **operator-** (const [move\\_iterator](#)< \_Iterator > &\_\_x, const [move\\_iterator](#)< \_Iterator > &\_\_y)-> decltype(\_\_x.base()-\_\_y.base())
- ptrdiff\_t **operator-** (const \_Bit\_iterator\_base &\_\_x, const \_Bit\_iterator\_base &\_\_y)
- template<typename \_Tp >  
[complex](#)< \_Tp > **operator-** (const [complex](#)< \_Tp > &\_\_x)
- template<typename \_Tp >  
\_Expr< \_BinClos< \_\_minus, \_ValArray, \_ValArray, \_Tp, \_Tp >, typename \_\_fun< \_\_minus, \_Tp >::result\_type > **operator-** (const [valarray](#)< \_Tp > &\_\_v, const [valarray](#)< \_Tp > &\_\_w)
- template<typename \_Tp >  
\_Expr< \_BinClos< \_\_minus, \_ValArray, \_Constant, \_Tp, \_Tp >, typename \_\_fun< \_\_minus, \_Tp >::result\_type > **operator-** (const [valarray](#)< \_Tp > &\_\_v, const \_Tp &\_\_t)
- template<typename \_Iterator >  
[reverse\\_iterator](#)< \_Iterator >::difference\_type **operator-** (const [reverse\\_iterator](#)< \_Iterator > &\_\_x, const [reverse\\_iterator](#)< \_Iterator > &\_\_y)
- template<typename \_IteratorL, typename \_IteratorR >  
auto **operator-** (const [move\\_iterator](#)< \_IteratorL > &\_\_x, const [move\\_iterator](#)< \_IteratorR > &\_\_y)-> decltype(\_\_x.base()-\_\_y.base())
- template<class \_Dom >  
\_Expr< \_BinClos< \_\_minus, \_ValArray, \_Expr, typename \_Dom::value\_type, \_Dom >, typename \_\_fun< \_\_minus, typename \_Dom::value\_type >::result\_type > **operator-** (const [valarray](#)< typename \_Dom::value\_type > &\_\_v, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e)
- template<typename \_Tp, typename \_Ref, typename \_Ptr >  
[Deque\\_iterator](#)< \_Tp, \_Ref, \_Ptr >::difference\_type **operator-** (const [Deque\\_iterator](#)< \_Tp, \_Ref, \_Ptr > &\_\_x, const [Deque\\_iterator](#)< \_Tp, \_Ref, \_Ptr > &\_\_y)
- template<class \_Dom >  
\_Expr< \_BinClos< \_\_minus, \_Expr, \_Constant, \_Dom, typename \_Dom::value\_type >, typename \_\_fun< \_\_minus, typename \_Dom::value\_type >::result\_type > **operator-** (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_v, const typename \_Dom::value\_type &\_\_t)

- `template<class _Dom >`  
`_Expr< _BinClos< __minus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __minus, typename _Dom::value_type >::result_type > operator- (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __minus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __minus, typename _Dom1::value_type >::result_type > operator- (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __minus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __minus, typename _Dom::value_type >::result_type > operator- (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __minus, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __minus, _Tp >::result_type > operator- (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _IteratorL, typename _IteratorR >`  
`auto operator- (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR > &__y)-> decltype(__y.base()-__x.base())`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`  
`Deque\_iterator< _Tp, _RefL, _PtrL >::difference_type operator- (const Deque\_iterator< _Tp, _RefL, _PtrL > &__x, const Deque\_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<class _Dom >`  
`_Expr< _BinClos< __divides, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __divides, typename _Dom::value_type >::result_type > operator/ (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __divides, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __divides, typename _Dom::value_type >::result_type > operator/ (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __divides, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __divides, _Tp >::result_type > operator/ (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __divides, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __divides, _Tp >::result_type > operator/ (const valarray< _Tp > &__v, const _Tp &__t)`



- `template<typename _Tp >`  
`_Expr< _BinClos< __divides, _Constant, _ValArray, _Tp, _Tp >, typename _`  
`_fun< __divides, _Tp >::result_type > operator/(const _Tp &__t, const valar-`  
`ray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __divides, _Constant, _Expr, typename _Dom::value_type,`  
`_Dom >, typename __fun< __divides, typename _Dom::value_type >::result_`  
`type > operator/(const typename _Dom::value_type &__t, const _Expr< _`  
`Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __divides, _Expr, _Constant, _Dom, typename _`  
`Dom::value_type >, typename __fun< __divides, typename _Dom::value_type`  
`>::result_type > operator/(const _Expr< _Dom, typename _Dom::value_type`  
`> &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1 , class _Dom2 >`  
`_Expr< _BinClos< __divides, _Expr, _Expr, _Dom1, _Dom2 >, typename`  
`__fun< __divides, typename _Dom1::value_type >::result_type > operator/`  
`(const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr<`  
`_Dom2, typename _Dom2::value_type > &__w)`
- `template<typename _Tp , typename _RefL , typename _PtrL , typename _RefR , typename _PtrR`  
`>`  
`bool operator<(const Deque\_iterator< _Tp, _RefL, _PtrL > &__x, const -`  
`Deque\_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename... _TElements, typename... _UElements>`  
`bool operator<(const tuple< _TElements...> &__t, const tuple< _`  
`_UElements...> &__u)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator<(const basic\_string< _CharT, _Traits, _Alloc > &__lhs, const`  
`basic\_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator<(const basic\_string< _CharT, _Traits, _Alloc > &__lhs, const`  
`_CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator<(const _CharT *__lhs, const basic\_string< _CharT, _Traits, _`  
`_Alloc > &__rhs)`
- `template<typename _Iterator >`  
`bool operator<(const reverse\_iterator< _Iterator > &__x, const reverse\_`  
`iterator< _Iterator > &__y)`
- `template<typename _Iterator >`  
`bool operator<(const move\_iterator< _Iterator > &__x, const move\_iterator<`  
`_Iterator > &__y)`
- `template<typename _Tp , typename _Alloc >`  
`bool operator<(const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc >`  
`&__y)`

- `template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >`  
`bool operator< (const map< _Key, _Tp, _Compare, _Alloc > &__x, const`  
`map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >`  
`bool operator< (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const`  
`multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key , typename _Compare , typename _Alloc >`  
`bool operator< (const multiset< _Key, _Compare, _Alloc > &__x, const multi-`  
`set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp , typename _Seq >`  
`bool operator< (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq >`  
`&__y)`
- `template<typename _Tp , typename _Alloc >`  
`bool operator< (const forward_list< _Tp, _Alloc > &__lx, const forward_list<`  
`_Tp, _Alloc > &__ly)`
- `template<typename _Key , typename _Compare , typename _Alloc >`  
`bool operator< (const set< _Key, _Compare, _Alloc > &__x, const set< _Key,`  
`_Compare, _Alloc > &__y)`
- `template<typename _Tp , typename _Seq >`  
`bool operator< (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq >`  
`&__y)`
- `template<typename _IteratorL , typename _IteratorR >`  
`bool operator< (const move_iterator< _IteratorL > &__x, const move_-`  
`iterator< _IteratorR > &__y)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less, _ValArray, _ValArray, _Tp, _Tp >, typename __-`  
`fun< __less, _Tp >::result_type > operator< (const valarray< _Tp > &__v,`  
`const valarray< _Tp > &__w)`
- `template<typename _Key , typename _Val , typename _KeyOfValue , typename _Compare , type-`  
`name _Alloc >`  
`bool operator< (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc`  
`> &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__-`  
`__y)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less, _ValArray, _Constant, _Tp, _Tp >, typename __-`  
`fun< __less, _Tp >::result_type > operator< (const valarray< _Tp > &__v,`  
`const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less, _Constant, _ValArray, _Tp, _Tp >, typename __-`  
`fun< __less, _Tp >::result_type > operator< (const _Tp &__t, const valarray<`  
`_Tp > &__v)`
- `template<typename _Tp , typename _Alloc >`  
`bool operator< (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc`  
`> &__y)`

- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`bool operator< (const \_Deque\_iterator< _Tp, _Ref, _Ptr > &__x, const \_Deque\_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`  
`bool operator< (const unique\_ptr< _Tp, _Dp > &__x, const unique\_ptr< _Up, _Ep > &__y)`
- `template<class _T1, class _T2 >`  
`constexpr bool operator< (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __less, typename _Dom::value_type >::result_type > operator< (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool operator< (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR > &__y)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __less, typename _Dom::value_type >::result_type > operator< (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __less, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __less, typename _Dom1::value_type >::result_type > operator< (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __less, typename _Dom::value_type >::result_type > operator< (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __less, typename _Dom::value_type >::result_type > operator< (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<typename _Tp, std::size_t _Nm>`  
`bool operator< (const array< _Tp, _Nm > &__a, const array< _Tp, _Nm > &__b)`
- `template<typename _Bilter >`  
`bool operator< (const sub\_match< _Bilter > &__lhs, const sub\_match< _Bilter > &__rhs)`

- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool operator< (const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`  
`bool operator< (const sub_match< _Bi_iter > &__lhs, const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`  
`bool operator< (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter >`  
`bool operator< (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator< (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b)`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`  
`bool operator< (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b)`
- `bool operator< (const error_code &__lhs, const error_code &__rhs)`
- `bool operator< (const error_condition &__lhs, const error_condition &__rhs)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &, const std::weibull_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &, const std::exponential_distribution< _RealType > &)`
- `template<class _CharT, class _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, thread::id __id)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const fisher_f_distribution< _RealType > &__x)`
- `template<typename _Tp, typename _CharT, class _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const complex< _Tp > &__x)`

- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _`  
`Traits > &__os, const basic_string< _CharT, _Traits, _Alloc > &__str)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _`  
`CharT, _Traits > &, const std::extreme_value_distribution< _RealType > &)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _`  
`CharT, _Traits > &__os, const discrete_distribution< _IntType > &__x)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template<typename, typename,`  
`typename > class _Base>`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _`  
`Traits > &__os, const __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _`  
`Base > &__str)`
- `template<typename _Tp>`  
`_Expr< _BinClos< __shift_left, _ValArray, _ValArray, _Tp, _Tp >, typename`  
`__fun< __shift_left, _Tp >::result_type > operator<< (const valarray< _Tp`  
`> &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp>`  
`_Expr< _BinClos< __shift_left, _ValArray, _Constant, _Tp, _Tp >, typename`  
`__fun< __shift_left, _Tp >::result_type > operator<< (const valarray< _Tp`  
`> &__v, const _Tp &__t)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename`  
`_Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _`  
`CharT, _Traits > &__os, const subtract_with_carry_engine< _UIntType, __w,`  
`__s, __r > &__x)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename`  
`_Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_`  
`ostream< _CharT, _Traits > &__os, const discard_block_engine< _`  
`RandomNumberEngine, __p, __r > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _`  
`CharT, _Traits > &, const std::cauchy_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream<`  
`_CharT, _Traits > &__os, const piecewise_linear_distribution< _RealType >`  
`&__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _`  
`CharT, _Traits > &__os, const chi_squared_distribution< _RealType > &__x)`
- `template<class _Dom>`  
`_Expr< _BinClos< __shift_left, _Expr, _ValArray, _Dom, typename _`

Dom::value\_type >, typename \_\_fun< \_\_shift\_left, typename Dom::value\_type >::result\_type > **operator**<< (const Expr< Dom, typename Dom::value\_type > &\_\_e, const [valarray](#)< typename Dom::value\_type > &\_\_v)

- template<typename RealType, typename CharT, typename Traits >  
[std::basic\\_ostream](#)< CharT, Traits > & **operator**<< ([std::basic\\_ostream](#)< CharT, Traits > &\_\_os, const [gamma\\_distribution](#)< RealType > &\_\_x)
- template<class Dom1, class Dom2 >  
Expr< BinClos< \_\_shift\_left, Expr, Expr, Dom1, Dom2 >, typename \_\_fun< \_\_shift\_left, typename Dom1::value\_type >::result\_type > **operator**<< (const Expr< Dom1, typename Dom1::value\_type > &\_\_v, const Expr< Dom2, typename Dom2::value\_type > &\_\_w)
- template<class Dom >  
Expr< BinClos< \_\_shift\_left, Expr, Constant, Dom, typename Dom::value\_type >, typename \_\_fun< \_\_shift\_left, typename Dom::value\_type >::result\_type > **operator**<< (const Expr< Dom, typename Dom::value\_type > &\_\_v, const typename Dom::value\_type &\_\_t)
- template<class Dom >  
Expr< BinClos< \_\_shift\_left, Constant, Expr, typename Dom::value\_type, Dom >, typename \_\_fun< \_\_shift\_left, typename Dom::value\_type >::result\_type > **operator**<< (const typename Dom::value\_type &\_\_t, const Expr< Dom, typename Dom::value\_type > &\_\_v)
- template<typename IntType, typename CharT, typename Traits >  
[std::basic\\_ostream](#)< CharT, Traits > & **operator**<< ([std::basic\\_ostream](#)< CharT, Traits > &\_\_os, const [binomial\\_distribution](#)< IntType > &\_\_x)
- template<typename RealType, typename CharT, typename Traits >  
[std::basic\\_ostream](#)< CharT, Traits > & **operator**<< ([std::basic\\_ostream](#)< CharT, Traits > &, const [std::uniform\\_real\\_distribution](#)< RealType > &)
- template<typename RealType, typename CharT, typename Traits >  
[std::basic\\_ostream](#)< CharT, Traits > & **operator**<< ([std::basic\\_ostream](#)< CharT, Traits > &\_\_os, const [lognormal\\_distribution](#)< RealType > &\_\_x)
- template<typename Tp >  
Expr< BinClos< \_\_shift\_left, Constant, ValArray, Tp, Tp >, typename \_\_fun< \_\_shift\_left, Tp >::result\_type > **operator**<< (const Tp &\_\_t, const [valarray](#)< Tp > &\_\_v)
- template<typename IntType, typename CharT, typename Traits >  
[std::basic\\_ostream](#)< CharT, Traits > & **operator**<< ([std::basic\\_ostream](#)< CharT, Traits > &, const [std::uniform\\_int\\_distribution](#)< IntType > &)
- template<typename UIntType, size\_t \_\_w, size\_t \_\_n, size\_t \_\_m, size\_t \_\_r, UIntType \_\_a, size\_t \_\_u, UIntType \_\_d, size\_t \_\_s, UIntType \_\_b, size\_t \_\_t, UIntType \_\_c, size\_t \_\_l, UIntType \_\_f, typename CharT, typename Traits >  
[std::basic\\_ostream](#)< CharT, Traits > & **operator**<< ([std::basic\\_ostream](#)< CharT, Traits > &\_\_os, const [mersenne\\_twister\\_engine](#)< UIntType, \_\_w, \_\_n, \_\_m, \_\_r, \_\_a, \_\_u, \_\_d, \_\_s, \_\_b, \_\_t, \_\_c, \_\_l, \_\_f > &\_\_x)

- `template<typename _IntType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream<  
_CharT, _Traits > &__os, const negative_binomial_distribution< _IntType >  
&__x)`
- `template<typename _CharT, typename _Traits >  
basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT,  
_Traits > &__os, _Resetiosflags __f)`
- `template<typename _CharT, typename _Traits >  
basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT,  
_Traits > &__os, _Setiosflags __f)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _  
CharT, _Traits > &__os, const piecewise_constant_distribution< _RealType >  
&__x)`
- `template<typename _CharT, typename _Traits >  
basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT,  
_Traits > &__os, _Setfill< _CharT > __f)`
- `template<typename _CharT, typename _Traits >  
basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT,  
_Traits > &__os, _Setprecision __f)`
- `template<typename _CharT, typename _Traits >  
basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT,  
_Traits > &__os, _Setw __f)`
- `template<typename _Ch_type, typename _Ch_traits, typename _Bi_iter >  
basic_ostream< _Ch_type, _Ch_traits > & operator<< (basic_ostream< _Ch_  
type, _Ch_traits > &__os, const sub_match< _Bi_iter > &__m)`
- `template<class _Dom >  
_Expr< _BinClos< __shift_left, _ValArray, _Expr, typename _Dom::value_  
type, _Dom >, typename __fun< __shift_left, typename _Dom::value_type  
>::result_type > operator<< (const valarray< typename _Dom::value_type >  
&__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp>  
std::basic_ostream< _Ch, _Tr > & operator<< (std::basic_ostream< _Ch, _Tr  
> &__os, const __shared_ptr< _Tp, _Lp > &__p)`
- `template<typename _CharT, typename _Traits, typename _MoneyT >  
basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT,  
_Traits > &__os, _Put_money< _MoneyT > __f)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _  
CharT, _Traits > &__os, const student_t_distribution< _RealType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _  
CharT, _Traits > &, const std::geometric_distribution< _IntType > &)`
- `template<typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _  
CharT, _Traits > &, const std::bernoulli_distribution &)`

- `template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const shuffle_order_engine< _RandomNumberEngine, __k > &__x)`
- `template<typename _CharT, typename _Traits, typename _Tp >  
basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &&__os, const _Tp &__x)`
- `template<typename _CharT, typename _Traits >  
basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, _Setbase __f)`
- `template<typename _IntType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const poisson_distribution< _IntType > &__x)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__x)`
- `template<typename _CharT, typename _Traits >  
basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const error_code &__e)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const linear_congruential_engine< _UIntType, __a, __c, __m > &__lcr)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const normal_distribution< _RealType > &__x)`
- `bool operator<= (thread::id __x, thread::id __y)`
- `template<typename... _TElements, typename... _UElements>  
bool operator<= (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<class _Dom >  
_Expr< _BinClos< __less_equal, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __less_equal, typename _Dom::value_type >::result_type > operator<= (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >  
bool operator<= (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Tp, std::size_t _Nm>  
bool operator<= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`



- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator<= (const _CharT *__lhs, const basic_string< _CharT, _Traits,`  
`_Alloc > &__rhs)`
- `template<typename _Iterator >`  
`bool operator<= (const reverse_iterator< _Iterator > &__x, const reverse_`  
`iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool operator<= (const move_iterator< _IteratorL > &__x, const move_`  
`iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`bool operator<= (const move_iterator< _Iterator > &__x, const move_`  
`iterator< _Iterator > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator<= (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc`  
`> &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator<= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc >`  
`&__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator<= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const`  
`map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator<= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x,`  
`const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool operator<= (const multiset< _Key, _Compare, _Alloc > &__x, const mul-`  
`tiset< _Key, _Compare, _Alloc > &__y)`
- `template<class _T1, class _T2 >`  
`constexpr bool operator<= (const pair< _T1, _T2 > &__x, const pair< _T1,`  
`_T2 > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool operator<= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq >`  
`&__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator<= (const forward_list< _Tp, _Alloc > &__lx, const forward_`  
`list< _Tp, _Alloc > &__ly)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool operator<= (const set< _Key, _Compare, _Alloc > &__x, const set< _`  
`Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool operator<= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq >`  
`&__y)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less_equal, _ValArray, _Constant, _Tp, _Tp >, typename`

- ```
__fun< __less_equal, _Tp >::result_type > operator<= (const valarray< _Tp
> &__v, const _Tp &__t)
```
- `template<typename _Tp, typename _Alloc >`
`bool operator<= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _`
`Alloc > &__y)`
 - `template<typename _IteratorL, typename _IteratorR >`
`bool operator<= (const reverse_iterator< _IteratorL > &__x, const reverse_`
`iterator< _IteratorR > &__y)`
 - `template<class _Dom >`
`_Expr< _BinClos< __less_equal, _ValArray, _Expr, typename _Dom::value_`
`type, _Dom >, typename __fun< __less_equal, typename _Dom::value_type`
`>::result_type > operator<= (const valarray< typename _Dom::value_type >`
`&__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
 - `template<typename _Tp >`
`_Expr< _BinClos< __less_equal, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __less_equal, _Tp >::result_type > operator<= (const valarray< _Tp`
`> &__v, const valarray< _Tp > &__w)`
 - `template<class _Dom >`
`_Expr< _BinClos< __less_equal, _Expr, _ValArray, _Dom, typename _`
`Dom::value_type >, typename __fun< __less_equal, typename _Dom::value_`
`type >::result_type > operator<= (const _Expr< _Dom, typename _`
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &_`
`__v)`
 - `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __less_equal, _Expr, _Expr, _Dom1, _Dom2 >, type-`
`name __fun< __less_equal, typename _Dom1::value_type >::result_type >`
`operator<= (const _Expr< _Dom1, typename _Dom1::value_type > &__v,`
`const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
 - `template<class _Dom >`
`_Expr< _BinClos< __less_equal, _Expr, _Constant, _Dom, typename _`
`Dom::value_type >, typename __fun< __less_equal, typename _Dom::value_`
`type >::result_type > operator<= (const _Expr< _Dom, typename _`
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
 - `template<typename _Bilter >`
`bool operator<= (const sub_match< _Bilter > &__lhs, const sub_match< _`
`Bilter > &__rhs)`
 - `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool operator<= (const basic_string< typename iterator_traits< _Bi_iter`
`>::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter >`
`&__rhs)`
 - `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool operator<= (const sub_match< _Bi_iter > &__lhs, const basic_string<`
`typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &_`
`__rhs)`

- `template<typename _Bi_iter >`
`bool operator<= (typename iterator_traits< _Bi_iter >::value_type const *__-`
`lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool operator<= (const sub_match< _Bi_iter > &__lhs, typename iterator-`
`traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter >`
`bool operator<= (typename iterator_traits< _Bi_iter >::value_type const &__-`
`lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool operator<= (const sub_match< _Bi_iter > &__lhs, typename iterator-`
`traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator<= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const`
`basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`bool operator<= (const Deque_iterator< _Tp, _Ref, _Ptr > &__x, const -`
`Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp >`
`_Expr< _BinClos< __less_equal, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __less_equal, _Tp >::result_type > operator<= (const _Tp &__t, const`
`valarray< _Tp > &__v)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator<= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const`
`_CharT *__rhs)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool operator<= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr<`
`_Up, _Ep > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR`
`>`
`bool operator<= (const Deque_iterator< _Tp, _RefL, _PtrL > &__x, const`
`Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _RealType >`
`bool operator== (const std::exponential_distribution< _RealType > &__d1,`
`const std::exponential_distribution< _RealType > &__d2)`
- `template<typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, bool >::__type op-`
`erator== (const basic_string< _CharT > &__lhs, const basic_string< _CharT`
`> &__rhs)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`
`bool operator== (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`
`&__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename... _TElements, typename... _UElements>`
`bool operator== (const tuple< _TElements...> &__t, const tuple< -`
`UElements...> &__u)`

- `template<typename _Tp, typename _Alloc >`
`bool operator== (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Iterator >`
`bool operator== (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _IntType >`
`bool operator== (const std::discrete_distribution< _IntType > &__d1, const std::discrete_distribution< _IntType > &__d2)`
- `template<typename _RealType >`
`bool operator== (const std::extreme_value_distribution< _RealType > &__d1, const std::extreme_value_distribution< _RealType > &__d2)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`
`bool operator== (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Iterator >`
`bool operator== (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator== (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `bool operator== (const std::bernoulli_distribution &__d1, const std::bernoulli_distribution &__d2)`
- `template<typename _CharT, typename _Traits >`
`bool operator== (const istreambuf_iterator< _CharT, _Traits > &__a, const istreambuf_iterator< _CharT, _Traits > &__b)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`bool operator== (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp >`
`_Expr< _BinClos< __equal_to, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __equal_to, _Tp >::result_type > operator== (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`bool operator== (const shared_ptr< _Tp > &__a, nullptr_t)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`bool operator== (const __unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__x, const __unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator== (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`

- `template<typename _Val >`
`bool operator== (const _List_iterator< _Val > &__x, const _List_const_iterator< _Val > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator== (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool operator== (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator== (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp >`
`_Expr< _BinClos< __equal_to, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __equal_to, _Tp >::result_type > operator== (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp, typename _Alloc >`
`bool operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp >`
`_Expr< _BinClos< __equal_to, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __equal_to, _Tp >::result_type > operator== (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp, typename _Seq >`
`bool operator== (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator== (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool operator== (nullptr_t, const unique_ptr< _Tp, _Dp > &__y)`
- `template<typename _Tp, typename _CharT, typename _Traits, typename _Dist >`
`bool operator== (const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__x, const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__y)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool operator== (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Biter >`
`bool operator== (const sub_match< _Biter > &__lhs, const sub_match< _Biter > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _Tp >`
`bool operator== (const allocator< _Tp > &, const allocator< _Tp > &)`

- `template<typename _Tp, typename _Seq >`
`bool operator== (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq >`
`&__y)`
- `template<typename _IntType >`
`bool operator== (const std::geometric_distribution< _IntType > &__d1, const`
`std::geometric_distribution< _IntType > &__d2)`
- `template<typename _RealType >`
`bool operator== (const std::piecewise_linear_distribution< _RealType > &__-`
`d1, const std::piecewise_linear_distribution< _RealType > &__d2)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, type-`
`name _Alloc >`
`bool operator== (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _-`
`Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc`
`> &__y)`
- `template<class _Dom >`
`_Expr< _BinClos< __equal_to, _Expr, _ValArray, _Dom, typename _-`
`Dom::value_type >, typename __fun< __equal_to, typename _Dom::value_-`
`type >::result_type > operator== (const _Expr< _Dom, typename _-`
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__-`
`v)`
- `template<class _T1, class _T2 >`
`constexpr bool operator== (const pair< _T1, _T2 > &__x, const pair< _T1,`
`_T2 > &__y)`
- `template<typename _StateT >`
`bool operator== (const fpos< _StateT > &__lhs, const fpos< _StateT > &__-`
`rhs)`
- `template<typename _Val >`
`bool operator== (const _Rb_tree_iterator< _Val > &__x, const _Rb_tree_-`
`const_iterator< _Val > &__y)`
- `template<typename _IntType >`
`bool operator== (const std::uniform_int_distribution< _IntType > &__d1, const`
`std::uniform_int_distribution< _IntType > &__d2)`
- `template<class _Dom >`
`_Expr< _BinClos< __equal_to, _Expr, _Constant, _Dom, typename _-`
`Dom::value_type >, typename __fun< __equal_to, typename _Dom::value_-`
`type >::result_type > operator== (const _Expr< _Dom, typename _-`
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __equal_to, _ValArray, _Expr, typename _Dom::value_-`
`type, _Dom >, typename __fun< __equal_to, typename _Dom::value_type`
`>::result_type > operator== (const valarray< typename _Dom::value_type >`
`&__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`
`bool operator== (const unordered_multiset< _Value, _Hash, _Pred, _Alloc >`
`&__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`

- `template<class _Value, class _Hash, class _Pred, class _Alloc >`
`bool operator== (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x,`
`const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, std::size_t _Nm>`
`bool operator== (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm`
`> &__two)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`bool operator== (const __unordered_set< _Value, _Hash, _Pred, _Alloc, __-`
`cache_hash_code > &__x, const __unordered_set< _Value, _Hash, _Pred, _-`
`Alloc, __cache_hash_code > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`bool operator== (const __unordered_multiset< _Value, _Hash, _Pred, _Alloc,`
`__cache_hash_code > &__x, const __unordered_multiset< _Value, _Hash, _-`
`Pred, _Alloc, __cache_hash_code > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool operator== (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`
`bool operator== (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _-`
`Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc`
`> &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash-`
`code>`
`bool operator== (const __unordered_multimap< _Key, _Tp, _Hash, _Pred, _-`
`Alloc, __cache_hash_code > &__x, const __unordered_multimap< _Key, _Tp,`
`_Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<typename _RealType >`
`bool operator== (const std::weibull_distribution< _RealType > &__d1, const`
`std::weibull_distribution< _RealType > &__d2)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __equal_to, _Expr, _Expr, _Dom1, _Dom2 >, typename _-`
`_fun< __equal_to, typename _Dom1::value_type >::result_type > operator==`
`(const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr<`
`_Dom2, typename _Dom2::value_type > &__w)`
- `template<typename _Res, typename... _Args>`
`bool operator== (const function< _Res(_Args...)> &__f, nullptr_t)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const`
`_CharT * __rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool operator== (const basic_string< typename iterator_traits< _Bi_iter`
`>::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter >`
`& __rhs)`
- `template<typename _Res, typename... _Args>`
`bool operator== (nullptr_t, const function< _Res(_Args...)> &__f)`

- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool operator== (const sub_match< _Bi_iter > &__lhs, const basic_string<`
`typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__`
`_rhs)`
- `template<typename _Bi_iter >`
`bool operator== (typename iterator_traits< _Bi_iter >::value_type const *__lhs,`
`const sub_match< _Bi_iter > &__rhs)`
- `template<class _Dom >`
`_Expr< _BinClos< __equal_to, _Constant, _Expr, typename _Dom::value_`
`type, _Dom >, typename __fun< __equal_to, typename _Dom::value_type`
`>::result_type > operator== (const typename _Dom::value_type &__t, const`
`_Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<typename _Bi_iter >`
`bool operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_`
`traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter >`
`bool operator== (typename iterator_traits< _Bi_iter >::value_type const &__`
`lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_`
`traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter, typename _Allocator >`
`bool operator== (const match_results< _Bi_iter, _Allocator > &__m1, const`
`match_results< _Bi_iter, _Allocator > &__m2)`
- `template<typename _RealType >`
`bool operator== (const std::cauchy_distribution< _RealType > &__d1, const`
`std::cauchy_distribution< _RealType > &__d2)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator== (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 >`
`&__b)`
- `template<typename _Tp >`
`bool operator== (nullptr_t, const shared_ptr< _Tp > &__b)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator== (const reverse_iterator< _IteratorL > &__x, const reverse_`
`iterator< _IteratorR > &__y)`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`
`bool operator== (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_`
`ptr< _Tp2, _Lp > &__b)`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool operator== (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t)`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool operator== (nullptr_t, const __shared_ptr< _Tp, _Lp > &__b)`
- `template<typename _T1, typename _T2 >`
`bool operator== (const allocator< _T1 > &, const allocator< _T2 > &)`

- `template<typename _RealType >`
`bool operator== (const std::normal_distribution< _RealType > &__d1, const std::normal_distribution< _RealType > &__d2)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator== (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _IntType >`
`bool operator== (const std::uniform_real_distribution< _IntType > &__d1, const std::uniform_real_distribution< _IntType > &__d2)`
- `template<typename _Tp >`
`bool operator== (const _Fwd_list_iterator< _Tp > &__x, const _Fwd_list_const_iterator< _Tp > &__y)`
- `bool operator== (const error_code &__lhs, const error_code &__rhs)`
- `template<typename _RealType >`
`bool operator== (const std::piecewise_constant_distribution< _RealType > &__d1, const std::piecewise_constant_distribution< _RealType > &__d2)`
- `bool operator== (const error_code &__lhs, const error_condition &__rhs)`
- `bool operator== (const error_condition &__lhs, const error_code &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool operator== (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `bool operator== (const error_condition &__lhs, const error_condition &__rhs)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __greater, _Tp >::result_type > operator> (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp, typename _Seq >`
`bool operator> (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator> (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `bool operator> (thread::id __x, thread::id __y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator> (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename... _TElements, typename... _UElements>`
`bool operator> (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`bool operator> (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y)`

- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator> (const _CharT *__lhs, const basic_string< _CharT, _Traits, _`
`Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const`
`basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Seq >`
`bool operator> (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq >`
`&__y)`
- `template<typename _Iterator >`
`bool operator> (const move_iterator< _Iterator > &__x, const move_iterator<`
`_Iterator > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool operator> (const map< _Key, _Tp, _Compare, _Alloc > &__x, const`
`map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool operator> (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const`
`multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __greater, _Tp >::result_type > operator> (const valarray< _Tp >`
`&__v, const _Tp &__t)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool operator> (const sub_match< _Bi_iter > &__lhs, const basic_string<`
`typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__`
`rhs)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool operator> (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up,`
`_Ep > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator> (const reverse_iterator< _IteratorL > &__x, const reverse_-`
`iterator< _IteratorR > &__y)`
- `template<class _Dom >`
`_Expr< _BinClos< __greater, _Expr, _ValArray, _Dom, typename _`
`Dom::value_type >, typename __fun< __greater, typename _Dom::value_type`
`>::result_type > operator> (const _Expr< _Dom, typename _Dom::value_-`
`type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _T1, class _T2 >`
`constexpr bool operator> (const pair< _T1, _T2 > &__x, const pair< _T1, _T2`
`> &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator> (const forward_list< _Tp, _Alloc > &__lx, const forward_list<`
`_Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`
`bool operator> (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc >`
`&__y)`

- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator> (const multiset< _Key, _Compare, _Alloc > &__x, const multi-`
`set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator> (const set< _Key, _Compare, _Alloc > &__x, const set< _Key,`
`_Compare, _Alloc > &__y)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __greater, _Expr, _Expr, _Dom1, _Dom2 >, typename _`
`_fun< __greater, typename _Dom1::value_type >::result_type > operator>`
`(const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr<`
`_Dom2, typename _Dom2::value_type > &__w)`
- `template<typename _Tp, typename _Alloc >`
`bool operator> (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc`
`> &__y)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater, _Constant, _ValArray, _Tp, _Tp >, typename _`
`_fun< __greater, _Tp >::result_type > operator> (const _Tp &__t, const valar-`
`ray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __greater, _Expr, _Constant, _Dom, typename _`
`Dom::value_type >, typename __fun< __greater, typename _Dom::value_type`
`>::result_type > operator> (const _Expr< _Dom, typename _Dom::value_`
`type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __greater, _ValArray, _Expr, typename _Dom::value_type,`
`_Dom >, typename __fun< __greater, typename _Dom::value_type >::result_`
`type > operator> (const valarray< typename _Dom::value_type > &__v, const`
`_Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< __greater, _Constant, _Expr, typename _Dom::value_type,`
`_Dom >, typename __fun< __greater, typename _Dom::value_type >::result_`
`type > operator> (const typename _Dom::value_type &__t, const _Expr< _`
`Dom, typename _Dom::value_type > &__v)`
- `template<typename _BiIter >`
`bool operator> (const sub_match< _BiIter > &__lhs, const sub_match< _`
`BiIter > &__rhs)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR`
`>`
`bool operator> (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _`
`Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, type-`
`name _Alloc >`
`bool operator> (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc`
`> &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__`
`__y)`

- `template<typename _Bi_iter >`
`bool operator> (typename iterator_traits< _Bi_iter >::value_type const *__lhs,`
`const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool operator> (const sub_match< _Bi_iter > &__lhs, typename iterator-`
`traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter >`
`bool operator> (const sub_match< _Bi_iter > &__lhs, typename iterator-`
`traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Tp, std::size_t _Nm>`
`bool operator> (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm >`
`&__two)`
- `template<typename _Iterator >`
`bool operator> (const reverse_iterator< _Iterator > &__x, const reverse-`
`iterator< _Iterator > &__y)`
- `template<typename _Bi_iter >`
`bool operator> (typename iterator_traits< _Bi_iter >::value_type const &__lhs,`
`const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool operator> (const basic_string< typename iterator_traits< _Bi_iter`
`>::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter >`
`&__rhs)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator>= (const move_iterator< _IteratorL > &__x, const move-`
`iterator< _IteratorR > &__y)`
- `bool operator>= (thread::id __x, thread::id __y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR`
`>`
`bool operator>= (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const`
`_Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc`
`> &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool operator>= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const`
`map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename... _TElements, typename... _UElements>`
`bool operator>= (const tuple< _TElements...> &__t, const tuple< _-`
`UElements...> &__u)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, type-`
`name _Alloc >`
`bool operator>= (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _-`
`Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc`
`> &__y)`

- `template<typename _Iterator >`
`bool operator>= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _Tp, std::size_t _Nm>`
`bool operator>= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Iterator >`
`bool operator>= (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool operator>= (const sub_match< _Bi_iter > &__lhs, const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`bool operator>= (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Bi_iter >`
`bool operator>= (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _BiIter >`
`bool operator>= (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator>= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Seq >`
`bool operator>= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool operator>= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater_equal, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __greater_equal, _Tp >::result_type > operator>= (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater_equal, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __greater_equal, _Tp >::result_type > operator>= (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp, typename _Seq >`
`bool operator>= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`

- `template<class _Dom >`
`_Expr< _BinClos< __greater_equal, _Expr, _ValArray, _Dom, typename`
`_Dom::value_type >, typename __fun< __greater_equal, typename _-`
`Dom::value_type >::result_type > operator>= (const _Expr< _Dom, type-`
`name _Dom::value_type > &__e, const valarray< typename _Dom::value_type`
`> &__v)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator>= (const multiset< _Key, _Compare, _Alloc > &__x, const mul-`
`tiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator>= (const set< _Key, _Compare, _Alloc > &__x, const set< _-`
`Key, _Compare, _Alloc > &__y)`
- `template<class _Dom >`
`_Expr< _BinClos< __greater_equal, _Expr, _Constant, _Dom, typename`
`_Dom::value_type >, typename __fun< __greater_equal, typename _-`
`Dom::value_type >::result_type > operator>= (const _Expr< _Dom, type-`
`name _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const forward_list< _Tp, _Alloc > &__lx, const forward_`
`list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _-`
`Alloc > &__y)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __greater_equal, _Expr, _Expr, _Dom1, _Dom2 >, type-`
`name __fun< __greater_equal, typename _Dom1::value_type >::result_type >`
`operator>= (const _Expr< _Dom1, typename _Dom1::value_type > &__v,`
`const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __greater_equal, _ValArray, _Expr, typename _-`
`Dom::value_type, _Dom >, typename __fun< __greater_equal, typename _-`
`Dom::value_type >::result_type > operator>= (const valarray< typename _-`
`Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type`
`> &__e)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater_equal, _ValArray, _ValArray, _Tp, _Tp >, type-`
`name __fun< __greater_equal, _Tp >::result_type > operator>= (const valar-`
`ray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __greater_equal, _Constant, _Expr, typename _-`
`Dom::value_type, _Dom >, typename __fun< __greater_equal, type-`
`name _Dom::value_type >::result_type > operator>= (const typename _-`
`Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type >`
`&__v)`

- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool operator>= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x,`
`const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Bi_iter >`
`bool operator>= (const sub_match< _Bi_iter > &__lhs, typename iterator_`
`traits< _Bi_iter >::value_type const *__rhs)`
- `template<class _T1, class _T2 >`
`constexpr bool operator>= (const pair< _T1, _T2 > &__x, const pair< _T1,`
`_T2 > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator>= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const`
`_CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator>= (const _CharT *__lhs, const basic_string< _CharT, _Traits,`
`_Alloc > &__rhs)`
- `template<typename _Bi_iter >`
`bool operator>= (const sub_match< _Bi_iter > &__lhs, typename iterator_`
`traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter >`
`bool operator>= (typename iterator_traits< _Bi_iter >::value_type const &__`
`lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool operator>= (const basic_string< typename iterator_traits< _Bi_iter`
`>::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter >`
`&__rhs)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator>= (const reverse_iterator< _IteratorL > &__x, const reverse_`
`iterator< _IteratorR > &__y)`
- `template<typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _`
`CharT, _Traits > &__is, std::bernoulli_distribution &__x)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a,`
`size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _`
`_UIntType __f, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _`
`CharT, _Traits > &__is, mersenne_twister_engine< _UIntType, __w, __n, __m,`
`__r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__x)`
- `template<typename _Tp, typename _CharT, class _Traits >`
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _`
`Traits > &__is, complex< _Tp > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _`
`CharT, _Traits > &, std::weibull_distribution< _RealType > &)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename`
`_Traits >`

[std::basic_istream](#)< _CharT, _Traits > & **operator**>> ([std::basic_istream](#)< _CharT, _Traits > &__is, [discard_block_engine](#)< _RandomNumberEngine, __p, __r > &__x)

- [template](#)<typename _RealType, typename _CharT, typename _Traits >
[std::basic_istream](#)< _CharT, _Traits > & **operator**>> ([std::basic_istream](#)< _CharT, _Traits > &__is, [chi_squared_distribution](#)< _RealType > &__x)
- [template](#)<typename _CharT, typename _Traits >
[basic_istream](#)< _CharT, _Traits > & **operator**>> ([basic_istream](#)< _CharT, _Traits > &__is, [_Setiosflags](#) __f)
- [template](#)<typename _RealType, typename _CharT, typename _Traits >
[std::basic_istream](#)< _CharT, _Traits > & **operator**>> ([std::basic_istream](#)< _CharT, _Traits > &__is, [student_t_distribution](#)< _RealType > &__x)
- [template](#)<typename _IntType, typename _CharT, typename _Traits >
[std::basic_istream](#)< _CharT, _Traits > & **operator**>> ([std::basic_istream](#)< _CharT, _Traits > &__is, [negative_binomial_distribution](#)< _IntType > &__x)
- [template](#)<typename _CharT, typename _Traits, typename _Alloc, [template](#)< typename, typename, typename > class _Base>
[basic_istream](#)< _CharT, _Traits > & **operator**>> ([basic_istream](#)< _CharT, _Traits > &__is, [__gnu_cxx::__versa_string](#)< _CharT, _Traits, _Alloc, _Base > &__str)
- [template](#)<typename _CharT, typename _Traits >
[basic_istream](#)< _CharT, _Traits > & **operator**>> ([basic_istream](#)< _CharT, _Traits > &__is, [_Setprecision](#) __f)
- [template](#)<typename _CharT, typename _Traits >
[basic_istream](#)< _CharT, _Traits > & **operator**>> ([basic_istream](#)< _CharT, _Traits > &__is, [_Setbase](#) __f)
- [template](#)<class _Dom >
_Expr< _BinClos< __shift_right, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __shift_right, typename _Dom::value_type >::result_type > **operator**>> (const [valarray](#)< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- [template](#)<>
[basic_istream](#)< char > & **operator**>> ([basic_istream](#)< char > &__is, [basic_string](#)< char > &__str)
- [template](#)<typename _RealType, typename _CharT, typename _Traits >
[std::basic_istream](#)< _CharT, _Traits > & **operator**>> ([std::basic_istream](#)< _CharT, _Traits > &__is, [normal_distribution](#)< _RealType > &__x)
- [template](#)<class _Dom >
_Expr< _BinClos< __shift_right, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __shift_right, typename _Dom::value_type >::result_type > **operator**>> (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- [template](#)<typename _RealType, typename _CharT, typename _Traits >
[std::basic_istream](#)< _CharT, _Traits > & **operator**>> ([std::basic_istream](#)< _CharT, _Traits > &, [std::cauchy_distribution](#)< _RealType > &)

- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _
CharT, _Traits > &__is, piecewise_constant_distribution< _RealType > &__x)`
- `template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _
CharT, _Traits > &__is, shuffle_order_engine< _RandomNumberEngine, __k
> &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _
CharT, _Traits > &__is, discrete_distribution< _IntType > &__x)`
- `template<typename _Tp >
_Expr< _BinClos< __shift_right, _Constant, _ValArray, _Tp, _Tp >, type-
name __fun< __shift_right, _Tp >::result_type > operator>> (const _Tp &_
_t, const valarray< _Tp > &__v)`
- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _
CharT, _Traits > &__is, gamma_distribution< _RealType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _
CharT, _Traits > &__is, binomial_distribution< _IntType > &__x)`
- `template<typename _Tp >
_Expr< _BinClos< __shift_right, _ValArray, _Constant, _Tp, _Tp >, typename
__fun< __shift_right, _Tp >::result_type > operator>> (const valarray< _Tp
> &__v, const _Tp &__t)`
- `template<class _Dom >
_Expr< _BinClos< __shift_right, _Expr, _Constant, _Dom, typename _
Dom::value_type >, typename __fun< __shift_right, typename _Dom::value_
type >::result_type > operator>> (const _Expr< _Dom, typename _
Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename
_Traits >
std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _
CharT, _Traits > &__is, subtract_with_carry_engine< _UIntType, __w, __s, __r
> &__x)`
- `template<typename _Tp >
_Expr< _BinClos< __shift_right, _ValArray, _ValArray, _Tp, _Tp >, typename
__fun< __shift_right, _Tp >::result_type > operator>> (const valarray< _Tp
> &__v, const valarray< _Tp > &__w)`
- `template<class _Dom1, class _Dom2 >
_Expr< _BinClos< __shift_right, _Expr, _Expr, _Dom1, _Dom2 >, type-
name __fun< __shift_right, typename _Dom1::value_type >::result_type >
operator>> (const _Expr< _Dom1, typename _Dom1::value_type > &__v,
const _Expr< _Dom2, typename _Dom2::value_type > &__w)`

- `template<typename _CharT, typename _Traits >`
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _`
`Traits > &__is, _Resetiosflags __f)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _`
`CharT, _Traits > &__is, piecewise_linear_distribution< _RealType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _`
`CharT, _Traits > &, std::geometric_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _`
`CharT, _Traits > &, std::exponential_distribution< _RealType > &)`
- `template<typename _CharT, typename _Traits >`
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _`
`Traits > &__is, _Setfill< _CharT > __f)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _`
`CharT, _Traits > &, std::extreme_value_distribution< _RealType > &)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_right, _Expr, _ValArray, _Dom, typename _`
`Dom::value_type >, typename __fun< __shift_right, typename _Dom::value_`
`type >::result_type > operator>> (const _Expr< _Dom, typename _`
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &_`
`_v)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _`
`CharT, _Traits > &__is, lognormal_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _`
`CharT, _Traits > &__is, fisher_f_distribution< _RealType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _`
`CharT, _Traits > &, std::uniform_int_distribution< _IntType > &)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _`
`Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str)`
- `template<typename _CharT, typename _Traits, typename _Tp >`
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _`
`Traits > &&__is, _Tp &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _`
`CharT, _Traits > &__is, poisson_distribution< _IntType > &__x)`
- `template<typename _CharT, typename _Traits, typename _MoneyT >`
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _`
`Traits > &__is, _Get_money< _MoneyT > __f)`

- `template<typename _CharT, typename _Traits >`
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > & __is, _Setw __f)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_real_distribution< _RealType > &)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is, linear_congruential_engine< _UIntType, __a, __c, __m > & __lcr)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __bitwise_xor, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __bitwise_xor, typename _Dom1::value_type >::result_type >`
`operator^ (const _Expr< _Dom1, typename _Dom1::value_type > & __v, const _Expr< _Dom2, typename _Dom2::value_type > & __w)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_xor, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __bitwise_xor, typename _Dom::value_type >::result_type >`
`operator^ (const valarray< typename _Dom::value_type > & __v, const _Expr< _Dom, typename _Dom::value_type > & __e)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_xor, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __bitwise_xor, _Tp >::result_type >`
`operator^ (const valarray< _Tp > & __v, const valarray< _Tp > & __w)`
- `constexpr _Ios_Iostate operator^ (_Ios_Iostate __a, _Ios_Iostate __b)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_xor, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __bitwise_xor, typename _Dom::value_type >::result_type >`
`operator^ (const _Expr< _Dom, typename _Dom::value_type > & __v, const typename _Dom::value_type & __t)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_xor, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __bitwise_xor, typename _Dom::value_type >::result_type >`
`operator^ (const _Expr< _Dom, typename _Dom::value_type > & __e, const valarray< typename _Dom::value_type > & __v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_xor, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __bitwise_xor, _Tp >::result_type >`
`operator^ (const _Tp & __t, const valarray< _Tp > & __v)`
- `constexpr _Ios_Openmode operator^ (_Ios_Openmode __a, _Ios_Openmode __b)`
- `constexpr _Ios_Fmtflags operator^ (_Ios_Fmtflags __a, _Ios_Fmtflags __b)`

- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_xor, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __bitwise_xor, typename _Dom::value_type >::result_type > operator^ (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_xor, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __bitwise_xor, _Tp >::result_type > operator^ (const valarray< _Tp > &__v, const _Tp &__t)`
- `const _Ios_Iostate & operator^= (_Ios_Iostate &__a, _Ios_Iostate __b)`
- `const _Ios_Openmode & operator^= (_Ios_Openmode &__a, _Ios_Openmode __b)`
- `const _Ios_Fmtflags & operator^= (_Ios_Fmtflags &__a, _Ios_Fmtflags __b)`
- `constexpr _Ios_Openmode operator| (_Ios_Openmode __a, _Ios_Openmode __b)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_or, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __bitwise_or, typename _Dom::value_type >::result_type > operator| (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_or, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __bitwise_or, typename _Dom::value_type >::result_type > operator| (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_or, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __bitwise_or, typename _Dom::value_type >::result_type > operator| (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_or, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __bitwise_or, _Tp >::result_type > operator| (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_or, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __bitwise_or, _Tp >::result_type > operator| (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `constexpr _Ios_Fmtflags operator| (_Ios_Fmtflags __a, _Ios_Fmtflags __b)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_or, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __bitwise_or, _Tp >::result_type > operator| (const _Tp &__t, const valarray< _Tp > &__v)`

- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __bitwise_or, _Expr, _Expr, _Dom1, _Dom2 >, type-`
`name __fun< __bitwise_or, typename _Dom1::value_type >::result_type >`
`operator| (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const`
`_Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_or, _Expr, _ValArray, _Dom, typename _-`
`Dom::value_type >, typename __fun< __bitwise_or, typename _Dom::value-`
`type >::result_type > operator| (const _Expr< _Dom, typename _-`
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__`
`_v)`
- `constexpr _Ios_Iostate operator| (_Ios_Iostate __a, _Ios_Iostate __b)`
- `const _Ios_Fmtflags & operator|= (_Ios_Fmtflags &__a, _Ios_Fmtflags __b)`
- `const _Ios_Openmode & operator|= (_Ios_Openmode &__a, _Ios_Openmode`
`__b)`
- `const _Ios_Iostate & operator|= (_Ios_Iostate &__a, _Ios_Iostate __b)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_or, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __logical_or, _Tp >::result_type > operator|| (const _Tp &__t, const`
`valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __logical_or, _ValArray, _Expr, typename _Dom::value-`
`type, _Dom >, typename __fun< __logical_or, typename _Dom::value_type`
`>::result_type > operator|| (const valarray< typename _Dom::value_type >`
`&__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< __logical_or, _Expr, _Constant, _Dom, typename _-`
`Dom::value_type >, typename __fun< __logical_or, typename _Dom::value-`
`type >::result_type > operator|| (const _Expr< _Dom, typename _-`
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __logical_or, _Expr, _Expr, _Dom1, _Dom2 >, type-`
`name __fun< __logical_or, typename _Dom1::value_type >::result_type >`
`operator|| (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const`
`_Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_or, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __logical_or, _Tp >::result_type > operator|| (const valarray< _Tp >`
`&__v, const valarray< _Tp > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __logical_or, _Constant, _Expr, typename _Dom::value-`
`type, _Dom >, typename __fun< __logical_or, typename _Dom::value_type`
`>::result_type > operator|| (const typename _Dom::value_type &__t, const _`
`Expr< _Dom, typename _Dom::value_type > &__v)`

- `template<typename _Tp >`
`_Expr< _BinClos< __logical_or, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __logical_or, _Tp >::result_type > operator||` (`const valarray< _Tp >`
`&__v, const _Tp &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __logical_or, _Expr, _ValArray, _Dom, typename _-`
`Dom::value_type >, typename __fun< __logical_or, typename _Dom::value-`
`type >::result_type > operator||` (`const _Expr< _Dom, typename _-`
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__-`
`_v)`
- `constexpr _Ios_Openmode operator~` (`_Ios_Openmode __a`)
- `constexpr _Ios_Fmtflags operator~` (`_Ios_Fmtflags __a`)
- `constexpr _Ios_Iostate operator~` (`_Ios_Iostate __a`)
- `template<typename _RAIter >`
`void partial_sort (_RAIter, _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`void partial_sort (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __-`
`middle, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __-`
`middle, _RandomAccessIterator __last)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare >`
`_RandomAccessIterator partial_sort_copy (_InputIterator __first, _InputIterator`
`__last, _RandomAccessIterator __result_first, _RandomAccessIterator __-`
`result_last, _Compare __comp)`
- `template<typename _Iter, typename _RAIter, typename _Compare >`
`_RAIter partial_sort_copy (_Iter, _Iter, _RAIter, _RAIter, _Compare)`
- `template<typename _Iter, typename _RAIter >`
`_RAIter partial_sort_copy (_Iter, _Iter, _RAIter, _RAIter)`
- `template<typename _InputIterator, typename _RandomAccessIterator >`
`_RandomAccessIterator partial_sort_copy (_InputIterator __first, _InputIterator`
`__last, _RandomAccessIterator __result_first, _RandomAccessIterator __-`
`result_last)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator partial_sum (_InputIterator __first, _InputIterator __last, _-`
`OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _InputIterator, typename _OutputIterator >`
`_OutputIterator partial_sum (_InputIterator __first, _InputIterator __last, _-`
`OutputIterator __result)`
- `template<typename _BIter, typename _Predicate >`
`_BIter partition (_BIter, _BIter, _Predicate)`

- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator partition (_ForwardIterator __first, _ForwardIterator __last, _-`
`Predicate __pred)`
- `template<typename _InputIterator, typename _OutputIterator1, typename _OutputIterator2, type-`
`name _Predicate >`
`pair< _OutputIterator1, _OutputIterator2 > partition_copy (_InputIterator __-`
`first, _InputIterator __last, _OutputIterator1 __out_true, _OutputIterator2 __-`
`out_false, _Predicate __pred)`
- `template<typename _Iter, typename _OIter1, typename _OIter2, typename _Predicate >`
`pair< _OIter1, _OIter2 > partition_copy (_Iter, _Iter, _OIter1, _OIter2, _-`
`Predicate)`
- `template<typename _Filter, typename _Predicate >`
`_Filter partition_point (_Filter, _Filter, _Predicate)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator partition_point (_ForwardIterator __first, _ForwardIterator __-`
`last, _Predicate __pred)`
- `template<typename _Tp >`
`complex< _Tp > polar (const _Tp &, const _Tp &=0)`
- `template<typename _RandomAccessIterator >`
`void pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last,`
`_Compare __comp)`
- `template<typename _RAIter, typename _Compare >`
`void pop_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`
`void pop_heap (_RAIter, _RAIter)`
- `template<typename _Tp >`
`complex< _Tp > pow (const complex< _Tp > &, const _Tp &)`
- `template<typename _Tp >`
`complex< _Tp > pow (const complex< _Tp > &, const complex< _Tp > &)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Pow, _Constant, _ValArray, _Tp, _Tp >, _Tp > pow`
`(const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp, typename _Up >`
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >`
`pow (const _Tp &__x, const std::complex< _Up > &__y)`
- `template<class _Dom >`
`_Expr< _BinClos< _Pow, _Expr, _Constant, _Dom, typename _Dom::value_-`
`type >, typename _Dom::value_type > pow (const _Expr< _Dom, typename`
`_Dom::value_type > &__e, const typename _Dom::value_type &__t)`
- `template<typename _Tp >`
`complex< _Tp > pow (const _Tp &, const complex< _Tp > &)`

- `template<typename _Tp >`
`_Expr< _BinClos< _Pow, _ValArray, _ValArray, _Tp, _Tp >, _Tp > pow`
`(const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Pow, _ValArray, _Constant, _Tp, _Tp >, _Tp > pow`
`(const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp, typename _Up >`
`__gnu_cxx::__promote_2< typename __gnu_cxx::__enable_if< __is_`
`arithmetic< _Tp >::__value &&__is_arithmetic< _Up >::__value, _Tp`
`>::__type, _Up >::__type pow (_Tp __x, _Up __y)`
- `long double pow (long double __x, long double __y)`
- `template<class _Dom >`
`_Expr< _BinClos< _Pow, _ValArray, _Expr, typename _Dom::value_type, _`
`Dom >, typename _Dom::value_type > pow (const valarray< typename _`
`Dom::valarray > &__v, const _Expr< _Dom, typename _Dom::value_type >`
`&__e)`
- `template<class _Dom >`
`_Expr< _BinClos< _Pow, _Expr, _ValArray, _Dom, typename _Dom::value_`
`type >, typename _Dom::value_type > pow (const _Expr< _Dom, typename _`
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &_`
`__v)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< _Pow, _Expr, _Expr, _Dom1, _Dom2 >, typename _`
`Dom1::value_type > pow (const _Expr< _Dom1, typename _Dom1::value_`
`type > &__e1, const _Expr< _Dom2, typename _Dom2::value_type > &__e2)`
- `template<class _Dom >`
`_Expr< _BinClos< _Pow, _Constant, _Expr, typename _Dom::value_type, _`
`Dom >, typename _Dom::value_type > pow (const typename _Dom::value_`
`type &__t, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp, typename _Up >`
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >`
`pow (const std::complex< _Tp > &__x, const _Up &__y)`
- `template<typename _Tp, typename _Up >`
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >`
`pow (const std::complex< _Tp > &__x, const std::complex< _Up > &__y)`
- `float pow (float __x, float __y)`
- `template<typename _BidirectionalIterator >`
`_BidirectionalIterator prev (_BidirectionalIterator __x, typename iterator_`
`traits< _BidirectionalIterator >::difference_type __n=1)`
- `template<typename _BidirectionalIterator >`
`bool prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator _`
`last)`

- `template<typename _BidirectionalIterator, typename _Compare >`
`bool prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __-`
`last, _Compare __comp)`
- `template<typename _BIter >`
`bool prev_permutation (_BIter, _BIter)`
- `template<typename _BIter, typename _Compare >`
`bool prev_permutation (_BIter, _BIter, _Compare)`
- `template<typename _Tp >`
`std::complex< _Tp > proj (const std::complex< _Tp > &)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type proj (_Tp __x)`
- `template<typename _Arg, typename _Result >`
`pointer_to_unary_function< _Arg, _Result > ptr_fun (_Result(*__x)(_Arg))`
- `template<typename _Arg1, typename _Arg2, typename _Result >`
`pointer_to_binary_function< _Arg1, _Arg2, _Result > ptr_fun (_Result(*__-`
`x)(_Arg1, _Arg2))`
- `template<typename _RAIter >`
`void push_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`void push_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator >`
`void push_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`
`last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void push_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`
`last, _Compare __comp)`
- `template<typename _MoneyT >`
`_Put_money< _MoneyT > put_money (const _MoneyT &__mon, bool __-`
`intl=false)`
- `template<typename _RandomAccessIterator >`
`void random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator`
`__last)`
- `template<typename _RAIter, typename _Generator >`
`void random_shuffle (_RAIter, _RAIter, _Generator &&)`
- `template<typename _RAIter >`
`void random_shuffle (_RAIter, _RAIter)`
- `template<typename _RandomAccessIterator, typename _RandomNumberGenerator >`
`void random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator`
`__last, _RandomNumberGenerator &&__rand)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type real (_Tp __x)`
- `template<typename _Tp >`
`constexpr _Tp real (const complex< _Tp > &__z)`

- `template<typename _ForwardIterator, typename _Tp >`
`_ForwardIterator remove (_ForwardIterator __first, _ForwardIterator __last,`
`const _Tp &__value)`
- `template<typename _Filter, typename _Tp >`
`_Filter remove (_Filter, _Filter, const _Tp &)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`
`_OutputIterator remove_copy (_InputIterator __first, _InputIterator __last, _-`
`OutputIterator __result, const _Tp &__value)`
- `template<typename _Iter, typename _OIter, typename _Tp >`
`_OIter remove_copy (_Iter, _Iter, _OIter, const _Tp &)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`
`_OutputIterator remove_copy_if (_InputIterator __first, _InputIterator __last, _-`
`OutputIterator __result, _Predicate __pred)`
- `template<typename _Iter, typename _OIter, typename _Predicate >`
`_OIter remove_copy_if (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator remove_if (_ForwardIterator __first, _ForwardIterator __last,`
`_Predicate __pred)`
- `template<typename _Filter, typename _Predicate >`
`_Filter remove_if (_Filter, _Filter, _Predicate)`
- `template<typename _Filter, typename _Tp >`
`void replace (_Filter, _Filter, const _Tp &, const _Tp &)`
- `template<typename _ForwardIterator, typename _Tp >`
`void replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__-`
`old_value, const _Tp &__new_value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`
`_OutputIterator replace_copy (_InputIterator __first, _InputIterator __last, _-`
`OutputIterator __result, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _Iter, typename _OIter, typename _Tp >`
`_OIter replace_copy (_Iter, _Iter, _OIter, const _Tp &, const _Tp &)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _-`
`Tp >`
`_OutputIterator replace_copy_if (_InputIterator __first, _InputIterator __last, _-`
`OutputIterator __result, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _Iter, typename _OIter, typename _Predicate, typename _Tp >`
`_OIter replace_copy_if (_Iter, _Iter, _OIter, _Predicate, const _Tp &)`
- `template<typename _ForwardIterator, typename _Predicate, typename _Tp >`
`void replace_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate _-`
`__pred, const _Tp &__new_value)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`
`void replace_if (_Filter, _Filter, _Predicate, const _Tp &)`
- `_Resetiosflags resetiosflags (ios_base::fmtflags __mask)`
- `void rethrow_exception (exception_ptr) __attribute__((__noreturn__))`
- `template<typename _Ex >`
`void rethrow_if_nested (const _Ex &__ex)`

- void [rethrow_if_nested](#) (const [nested_exception](#) &__ex)
- template<typename _Tp >
void [return_temporary_buffer](#) (_Tp *__p)
- template<typename _BidirectionalIterator >
void [reverse](#) (_BidirectionalIterator __first, _BidirectionalIterator __last)
- template<typename _BIter >
void [reverse](#) (_BIter, _BIter)
- template<typename _BIter, typename _OIter >
_OIter [reverse_copy](#) (_BIter, _BIter, _OIter)
- template<typename _BidirectionalIterator, typename _OutputIterator >
_OutputIterator [reverse_copy](#) (_BidirectionalIterator __first, _BidirectionalIterator __last, _OutputIterator __result)
- [ios_base](#) & [right](#) ([ios_base](#) &__base)
- template<typename _Filter >
void [rotate](#) (_Filter, _Filter, _Filter)
- template<typename _ForwardIterator >
void [rotate](#) (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last)
- template<typename _Filter, typename _OIter >
_OIter [rotate_copy](#) (_Filter, _Filter, _Filter, _OIter)
- template<typename _ForwardIterator, typename _OutputIterator >
_OutputIterator [rotate_copy](#) (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, _OutputIterator __result)
- [ios_base](#) & [scientific](#) ([ios_base](#) &__base)
- template<typename _ForwardIterator1, typename _ForwardIterator2 >
_ForwardIterator1 [search](#) (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)
- template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >
_ForwardIterator1 [search](#) (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate)
- template<typename _Filter1, typename _Filter2 >
_Filter1 [search](#) (_Filter1, _Filter1, _Filter2, _Filter2)
- template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >
_Filter1 [search](#) (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)
- template<typename _Filter, typename _Size, typename _Tp, typename _BinaryPredicate >
_Filter [search_n](#) (_Filter, _Filter, _Size, const _Tp &, _BinaryPredicate)
- template<typename _Filter, typename _Size, typename _Tp >
_Filter [search_n](#) (_Filter, _Filter, _Size, const _Tp &)
- template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >
_ForwardIterator [search_n](#) (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)

- `template<typename _ForwardIterator, typename _Integer, typename _Tp >`
`_ForwardIterator search_n (_ForwardIterator __first, _ForwardIterator __last, _-`
`Integer __count, const _Tp &__val)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`
`name _Compare >`
`_OutputIterator set_difference (_InputIterator1 __first1, _InputIterator1 __last1,`
`_InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _-`
`Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator set_difference (_InputIterator1 __first1, _InputIterator1 __last1,`
`_InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator set_intersection (_InputIterator1 __first1, _InputIterator1 __-`
`last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`
`name _Compare >`
`_OutputIterator set_intersection (_InputIterator1 __first1, _InputIterator1 __-`
`last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result,`
`_Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `new_handler set_new_handler (new_handler) throw ()`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator,`
`typename _Compare >`
`_OutputIterator set_symmetric_difference (_InputIterator1 __first1, _-`
`InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2,`
`_OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator set_symmetric_difference (_InputIterator1 __first1, _-`
`InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2,`
`_OutputIterator __result)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _-`
`Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `terminate_handler set_terminate (terminate_handler) throw ()`
- `unexpected_handler set_unexpected (unexpected_handler) throw ()`

- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator set_union (_InputIterator1 __first1, _InputIterator1 __last1, _-`
`InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`
`name _Compare >`
`_OutputIterator set_union (_InputIterator1 __first1, _InputIterator1 __last1, _-`
`InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _-`
`Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `_Setbase setbase (int __base)`
- `template<typename _CharT >`
`_Setfill< _CharT > setfill (_CharT __c)`
- `_Setiosflags setiosflags (ios_base::fmtflags __mask)`
- `_Setprecision setprecision (int __n)`
- `_Setw setw (int __n)`
- `ios_base & showbase (ios_base &__base)`
- `ios_base & showpoint (ios_base &__base)`
- `ios_base & showpos (ios_base &__base)`
- `template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator >`
`void shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _-`
`UniformRandomNumberGenerator &&__g)`
- `template<typename _RAIter, typename _UGenerator >`
`void shuffle (_RAIter, _RAIter, _UGenerator &&)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Sin, _ValArray, _Tp >, _Tp > sin (const valarray< _Tp >`
`&__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Sin, _Expr, _Dom >, typename _Dom::value_type > sin`
`(const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type sin`
`(_Tp __x)`
- `template<typename _Tp >`
`complex< _Tp > sin (const complex< _Tp > &)`
- `float sin (float __x)`
- `long double sin (long double __x)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`
`sinh (_Tp __x)`
- `float sinh (float __x)`

- `template<typename _Tp >`
`_Expr< _UnClos< _Sinh, _ValArray, _Tp >, _Tp > sinh (const valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Sinh, _Expr, _Dom >, typename _Dom::value_type > sinh (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`complex< _Tp > sinh (const complex< _Tp > &)`
- `long double sinh (long double __x)`
- `ios_base & skipws (ios_base &__base)`
- `template<typename _RAIter, typename _Compare >`
`void sort (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`
`void sort (_RAIter, _RAIter)`
- `template<typename _RandomAccessIterator >`
`void sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RAIter, typename _Compare >`
`void sort_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`
`void sort_heap (_RAIter, _RAIter)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _Tp >`
`complex< _Tp > sqrt (const complex< _Tp > &)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type sqrt (_Tp __x)`
- `float sqrt (float __x)`
- `long double sqrt (long double __x)`
- `template<class _Dom >`
`_Expr< _UnClos< _Sqrt, _Expr, _Dom >, typename _Dom::value_type > sqrt (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Sqrt, _ValArray, _Tp >, _Tp > sqrt (const valarray< _Tp > &__v)`
- `template<typename _BIter, typename _Predicate >`
`_BIter stable_partition (_BIter, _BIter, _Predicate)`

- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator stable_partition (_ForwardIterator __first, _ForwardIterator __-`
`__last, _Predicate __pred)`
- `template<typename _RAIter, typename _Compare >`
`void stable_sort (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __-`
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __-`
`__last)`
- `template<typename _RAIter >`
`void stable_sort (_RAIter, _RAIter)`
- `template<typename _Tp, typename _Tp1 >`
`shared_ptr< _Tp > static_pointer_cast (const shared_ptr< _Tp1 > &__r)`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>`
`__shared_ptr< _Tp, _Lp > static_pointer_cast (const __shared_ptr< _Tp1, _Lp`
`> &__r)`
- `double stod (const string &__str, size_t *__idx=0)`
- `float stof (const string &__str, size_t *__idx=0)`
- `int stoi (const string &__str, size_t *__idx=0, int __base=10)`
- `long stol (const string &__str, size_t *__idx=0, int __base=10)`
- `long double stold (const string &__str, size_t *__idx=0)`
- `long long stoll (const string &__str, size_t *__idx=0, int __base=10)`
- `unsigned long stoul (const string &__str, size_t *__idx=0, int __base=10)`
- `unsigned long long stoull (const string &__str, size_t *__idx=0, int __base=10)`
- `char * strchr (char *__s, int __n)`
- `char * strpbrk (char *__s1, const char *__s2)`
- `char * strrchr (char *__s, int __n)`
- `char * strstr (char *__s1, const char *__s2)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`void swap (map< _Key, _Tp, _Compare, _Alloc > &__x, map< _Key, _Tp,`
`_Compare, _Alloc > &__y)`
- `template<typename _Res, typename... _Args>`
`void swap (function< _Res(_Args...)> &__x, function< _Res(_Args...)> &__-`
`y)`
- `void swap (thread &__x, thread &__y)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, type-`
`name _Alloc >`
`void swap (_Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x,`
`_Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Tp >`
`void swap (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b)`

- `template<typename _Res, typename... _ArgTypes>`
`void swap (packaged_task< _Res(_ArgTypes...)> &__x, packaged_task< _Res(_ArgTypes...)> &__y)`
- `template<class _T1, class _T2 >`
`void swap (pair< _T1, _T2 > &__x, pair< _T1, _T2 > &__y)`
- `template<typename _Tp, typename _Dp >`
`void swap (unique_ptr< _Tp, _Dp > &__x, unique_ptr< _Tp, _Dp > &__y)`
- `template<typename _Tp >`
`void swap (_Tp &__a, _Tp &__b)`
- `template<typename _Tp, typename _Seq >`
`void swap (queue< _Tp, _Seq > &__x, queue< _Tp, _Seq > &__y)`
- `template<typename _Tp >`
`void swap (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`void swap (__unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__x, __unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<typename _Tp, typename _Alloc >`
`void swap (list< _Tp, _Alloc > &__x, list< _Tp, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`void swap (__unordered_set< _Value, _Hash, _Pred, _Alloc, __cache_hash_code > &__x, __unordered_set< _Value, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<typename _Bi_iter, typename _Allocator >`
`void swap (match_results< _Bi_iter, _Allocator > &__lhs, match_results< _Bi_iter, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Sequence, typename _Compare >`
`void swap (priority_queue< _Tp, _Sequence, _Compare > &__x, priority_queue< _Tp, _Sequence, _Compare > &__y)`
- `template<typename _Tp, _Lock_policy _Lp>`
`void swap (__weak_ptr< _Tp, _Lp > &__a, __weak_ptr< _Tp, _Lp > &__b)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`
`void swap (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Res >`
`void swap (promise< _Res > &__x, promise< _Res > &__y)`
- `template<typename _Tp, typename _Alloc >`
`void swap (vector< _Tp, _Alloc > &__x, vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, size_t _Nm>`
`void swap (_Tp(&)[_Nm], _Tp(&)[_Nm])`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`void swap (multiset< _Key, _Compare, _Alloc > &__x, multiset< _Key, _Compare, _Alloc > &__y)`

- `template<typename _Ch_type, typename _Rx_traits >`
`void swap (basic_regex< _Ch_type, _Rx_traits > &__lhs, basic_regex< _Ch_type, _Rx_traits > &__rhs)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`
`void swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Seq >`
`void swap (stack< _Tp, _Seq > &__x, stack< _Tp, _Seq > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`void swap (__unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__x, __unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`void swap (set< _Key, _Compare, _Alloc > &__x, set< _Key, _Compare, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`void swap (__unordered_multiset< _Value, _Hash, _Pred, _Alloc, __cache_hash_code > &__x, __unordered_multiset< _Value, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<typename _Tp, typename _Alloc >`
`void swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`
`void swap (deque< _Tp, _Alloc > &__x, deque< _Tp, _Alloc > &__y)`
- `template<typename _Mutex >`
`void swap (unique_lock< _Mutex > &__x, unique_lock< _Mutex > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`
`void swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, std::size_t _Nm>`
`void swap (array< _Tp, _Nm > &__one, array< _Tp, _Nm > &__two)`
- `template<typename... _Elements>`
`void swap (tuple< _Elements...> &__x, tuple< _Elements...> &__y)`
- `template<typename _Tp, _Lock_policy _Lp>`
`void swap (__shared_ptr< _Tp, _Lp > &__a, __shared_ptr< _Tp, _Lp > &__b)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`
`void swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`void swap (multimap< _Key, _Tp, _Compare, _Alloc > &__x, multimap< _Key, _Tp, _Compare, _Alloc > &__y)`

- `template<typename _CharT, typename _Traits, typename _Alloc >`
`void swap (basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string< _`
`CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _Filter1, typename _Filter2 >`
`_Filter2 swap_ranges (_Filter1, _Filter1, _Filter2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`_ForwardIterator2 swap_ranges (_ForwardIterator1 __first1, _ForwardIterator1`
`__last1, _ForwardIterator2 __first2)`
- `const error_category & system_category () throw ()`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type tan`
`(_Tp __x)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Tan, _ValArray, _Tp >, _Tp > tan (const valarray< _Tp`
`> &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Tan, _Expr, _Dom >, typename _Dom::value_type > tan`
`(const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `float tan (float __x)`
- `template<typename _Tp >`
`complex< _Tp > tan (const complex< _Tp > &)`
- `long double tan (long double __x)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Tanh, _ValArray, _Tp >, _Tp > tanh (const valarray< _Tp`
`> &__v)`
- `long double tanh (long double __x)`
- `template<class _Dom >`
`_Expr< _UnClos< _Tanh, _Expr, _Dom >, typename _Dom::value_type >`
`tanh (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`
`tanh (_Tp __x)`
- `float tanh (float __x)`
- `template<typename _Tp >`
`complex< _Tp > tanh (const complex< _Tp > &)`
- `void terminate () __attribute__((__noreturn__)) throw ()`
- `template<typename _Ex >`
`void throw_with_nested (_Ex __ex)`
- `template<typename... _Elements>`
`tuple< _Elements &...> tie (_Elements &...__args)`
- `string to_string (unsigned __val)`
- `string to_string (long __val)`
- `string to_string (long double __val)`
- `string to_string (long long __val)`

- `string to_string` (int __val)
- `string to_string` (float __val)
- `string to_string` (unsigned long long __val)
- `string to_string` (unsigned long __val)
- `string to_string` (double __val)
- `wstring to_wstring` (long __val)
- `wstring to_wstring` (int __val)
- `wstring to_wstring` (unsigned __val)
- `wstring to_wstring` (float __val)
- `wstring to_wstring` (unsigned long __val)
- `wstring to_wstring` (unsigned long long __val)
- `wstring to_wstring` (long double __val)
- `wstring to_wstring` (double __val)
- `wstring to_wstring` (long long __val)
- `template<typename _CharT >`
`_CharT tolower` (_CharT __c, const `locale` &__loc)
- `template<typename _CharT >`
`_CharT toupper` (_CharT __c, const `locale` &__loc)
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BinaryOperation >`
`_OIter transform` (_Iter1, _Iter1, _Iter2, _OIter, _BinaryOperation)
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`
`_OIter transform` (_Iter, _Iter, _OIter, _UnaryOperation)
- `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation >`
`_OutputIterator transform` (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _UnaryOperation __unary_op)
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator transform` (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _OutputIterator __result, _BinaryOperation __binary_op)
- `template<typename _Lock1, typename _Lock2, typename... _Lock3>`
`int try_lock` (_Lock1 &__l1, _Lock2 &__l2, _Lock3 &...__l3)
- `template<typename... _TElements, typename... _UElements>`
`tuple< _TElements..., _UElements...> tuple_cat` (tuple< _TElements...> &&__t, tuple< _UElements...> &&__u)
- `template<typename... _TElements, typename... _UElements>`
`tuple< _TElements..., _UElements...> tuple_cat` (const tuple< _TElements...> &&__t, tuple< _UElements...> &&__u)
- `template<typename... _TElements, typename... _UElements>`
`tuple< _TElements..., _UElements...> tuple_cat` (tuple< _TElements...> &&__t, const tuple< _UElements...> &__u)
- `template<typename... _TElements, typename... _UElements>`
`tuple< _TElements..., _UElements...> tuple_cat` (const tuple< _TElements...> &&__t, const tuple< _UElements...> &__u)

- bool [uncaught_exception](#) () __attribute__((__pure__)) throw ()
- void [unexpected](#) () __attribute__((__noreturn__))
- template<typename _InputIterator, typename _ForwardIterator >
_ForwardIterator [uninitialized_copy](#) (_InputIterator __first, _InputIterator __last, _ForwardIterator __result)
- template<typename _InputIterator, typename _Size, typename _ForwardIterator >
_ForwardIterator [uninitialized_copy_n](#) (_InputIterator __first, _Size __n, _ForwardIterator __result)
- template<typename _ForwardIterator, typename _Tp >
void [uninitialized_fill](#) (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__x)
- template<typename _ForwardIterator, typename _Size, typename _Tp >
void [uninitialized_fill_n](#) (_ForwardIterator __first, _Size __n, const _Tp &__x)
- template<typename _FIter, typename _BinaryPredicate >
_FIter [unique](#) (_FIter, _FIter, _BinaryPredicate)
- template<typename _ForwardIterator, typename _BinaryPredicate >
_ForwardIterator [unique](#) (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)
- template<typename _ForwardIterator >
_ForwardIterator [unique](#) (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _FIter >
_FIter [unique](#) (_FIter, _FIter)
- template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate >
_OutputIterator [unique_copy](#) (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred)
- template<typename _InputIterator, typename _OutputIterator >
_OutputIterator [unique_copy](#) (_InputIterator __first, _InputIterator __last, _OutputIterator __result)
- template<typename _IIter, typename _OIter, typename _BinaryPredicate >
_OIter [unique_copy](#) (_IIter, _IIter, _OIter, _BinaryPredicate)
- template<typename _IIter, typename _OIter >
_OIter [unique_copy](#) (_IIter, _IIter, _OIter)
- [ios_base](#) & [unitbuf](#) ([ios_base](#) &__base)
- template<typename _ForwardIterator, typename _Tp >
_ForwardIterator [upper_bound](#) (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)
- template<typename _ForwardIterator, typename _Tp, typename _Compare >
_ForwardIterator [upper_bound](#) (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)
- template<typename _FIter, typename _Tp >
_FIter [upper_bound](#) (_FIter, _FIter, const _Tp &)
- template<typename _FIter, typename _Tp, typename _Compare >
_FIter [upper_bound](#) (_FIter, _FIter, const _Tp &, _Compare)
- [ios_base](#) & [uppercase](#) ([ios_base](#) &__base)

- `template<typename _Facet >`
`const _Facet & use_facet (const locale & __loc)`
- `wchar_t * wcsrchr (wchar_t * __p, wchar_t __c)`
- `wchar_t * wcsprbrk (wchar_t * __s1, const wchar_t * __s2)`
- `wchar_t * wcsrchr (wchar_t * __p, wchar_t __c)`
- `wchar_t * wcsstr (wchar_t * __s1, const wchar_t * __s2)`
- `wchar_t * wmemchr (wchar_t * __p, wchar_t __c, size_t __n)`
- `template<typename _CharT, typename _Traits >`
`basic_istream< _CharT, _Traits > & ws (basic_istream< _CharT, _Traits > & __is)`
- `template<size_t _Nb>`
`bitset< _Nb > operator& (const bitset< _Nb > & __x, const bitset< _Nb > & __y)`
- `template<size_t _Nb>`
`bitset< _Nb > operator| (const bitset< _Nb > & __x, const bitset< _Nb > & __y)`
- `template<size_t _Nb>`
`bitset< _Nb > operator^ (const bitset< _Nb > & __x, const bitset< _Nb > & __y)`
- `template<class _CharT, class _Traits, size_t _Nb>`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is, bitset< _Nb > & __x)`
- `template<class _CharT, class _Traits, size_t _Nb>`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > & __os, const bitset< _Nb > & __x)`
- `template<typename _Tp >`
`complex< _Tp > operator+ (const complex< _Tp > & __x, const complex< _Tp > & __y)`
- `template<typename _Tp >`
`complex< _Tp > operator+ (const complex< _Tp > & __x, const _Tp & __y)`
- `template<typename _Tp >`
`complex< _Tp > operator+ (const _Tp & __x, const complex< _Tp > & __y)`
- `template<typename _Tp >`
`complex< _Tp > operator- (const complex< _Tp > & __x, const complex< _Tp > & __y)`
- `template<typename _Tp >`
`complex< _Tp > operator- (const complex< _Tp > & __x, const _Tp & __y)`
- `template<typename _Tp >`
`complex< _Tp > operator- (const _Tp & __x, const complex< _Tp > & __y)`

- `template<typename _Tp >`
`complex<_Tp> operator* (const complex<_Tp> &__x, const complex<_Tp> &__y)`
- `template<typename _Tp >`
`complex<_Tp> operator* (const complex<_Tp> &__x, const _Tp &__y)`
- `template<typename _Tp >`
`complex<_Tp> operator* (const _Tp &__x, const complex<_Tp> &__y)`

- `template<typename _Tp >`
`complex<_Tp> operator/ (const complex<_Tp> &__x, const complex<_Tp> &__y)`
- `template<typename _Tp >`
`complex<_Tp> operator/ (const complex<_Tp> &__x, const _Tp &__y)`
- `template<typename _Tp >`
`complex<_Tp> operator/ (const _Tp &__x, const complex<_Tp> &__y)`

- `template<typename _Tp >`
`constexpr bool operator== (const complex<_Tp> &__x, const complex<_Tp> &__y)`
- `template<typename _Tp >`
`constexpr bool operator== (const complex<_Tp> &__x, const _Tp &__y)`
- `template<typename _Tp >`
`constexpr bool operator== (const _Tp &__x, const complex<_Tp> &__y)`

- `template<typename _Tp >`
`constexpr bool operator!= (const complex<_Tp> &__x, const complex<_Tp> &__y)`
- `template<typename _Tp >`
`constexpr bool operator!= (const complex<_Tp> &__x, const _Tp &__y)`
- `template<typename _Tp >`
`constexpr bool operator!= (const _Tp &__x, const complex<_Tp> &__y)`

- `template<typename _Tp >`
`reference_wrapper<_Tp> ref (_Tp &__t)`
- `template<typename _Tp >`
`reference_wrapper<const _Tp> cref (const _Tp &__t)`
- `template<typename _Tp >`
`reference_wrapper<_Tp> ref (reference_wrapper<_Tp> __t)`
- `template<typename _Tp >`
`reference_wrapper<const _Tp> cref (reference_wrapper<_Tp> __t)`

- `template<typename _CharT, typename _Traits >`
`basic_istream<_CharT, _Traits> & operator>> (basic_istream<_CharT, _Traits> &__in, _CharT &__c)`
- `template<class _Traits >`
`basic_istream<char, _Traits> & operator>> (basic_istream<char, _Traits> &__in, unsigned char &__c)`

- `template<class _Traits >`
`basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits`
`> &__in, signed char &__c)`
- `template<typename _CharT, typename _Traits >`
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT,`
`_Traits > &__in, _CharT *__s)`
- `template<>`
`basic_istream< char > & operator>> (basic_istream< char > &__in, char`
`*__s)`
- `template<class _Traits >`
`basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits`
`> &__in, unsigned char *__s)`
- `template<class _Traits >`
`basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits`
`> &__in, signed char *__s)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT,`
`_Traits > &__out, _CharT __c)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT,`
`_Traits > &__out, char __c)`
- `template<class _Traits >`
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits`
`> &__out, char __c)`
- `template<class _Traits >`
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits`
`> &__out, signed char __c)`
- `template<class _Traits >`
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits`
`> &__out, unsigned char __c)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT,`
`_Traits > &__out, const _CharT *__s)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT,`
`_Traits > &__out, const char *__s)`
- `template<class _Traits >`
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits`
`> &__out, const char *__s)`
- `template<class _Traits >`
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits`
`> &__out, const signed char *__s)`

- `template<class _Traits >`
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits`
`> &__out, const unsigned char *__s)`

Matching, Searching, and Replacing

- `template<typename _Bi_iter, typename _Allocator, typename _Ch_type, typename _Rx_traits`
`>`
`bool regex_match (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _`
`_Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re,`
`regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >`
`bool regex_match (_Bi_iter __first, _Bi_iter __last, const basic_regex<`
`_Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __`
`flags=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Allocator, typename _Rx_traits >`
`bool regex_match (const _Ch_type *__s, match_results< const _Ch_type *,`
`_Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re,`
`regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Allocator, typename _Ch_`
`type, typename _Rx_traits >`
`bool regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc >`
`&__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _`
`_Ch_alloc >::const_iterator, _Allocator > &__m, const basic_regex< _Ch_`
`type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_`
`constants::match_default)`
- `template<typename _Ch_type, class _Rx_traits >`
`bool regex_match (const _Ch_type *__s, const basic_regex< _Ch_`
`type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_`
`constants::match_default)`
- `template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _`
`Rx_traits >`
`bool regex_match (const basic_string< _Ch_type, _Ch_traits, _Str_allocator`
`> &__s, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_`
`constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Allocator, typename _Ch_type, typename _Rx_traits`
`>`
`bool regex_search (_Bi_iter __first, _Bi_iter __last, match_results< _Bi_`
`iter, _Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re,`
`regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >`
`bool regex_search (_Bi_iter __first, _Bi_iter __last, const basic_regex<`
`_Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __`
`flags=regex_constants::match_default)`
- `template<typename _Ch_type, class _Allocator, class _Rx_traits >`
`bool regex_search (const _Ch_type *__s, match_results< const _Ch_type`


```

*, _Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits > &__e,
regex_constants::match_flag_type __f=regex_constants::match_default)
• template<typename _Ch_type, typename _Rx_traits >
bool regex_search (const _Ch_type *__s, const basic_regex< _Ch_
type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_
constants::match_default)
• template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename
_Rx_traits >
bool regex_search (const basic_string< _Ch_type, _Ch_traits, _String_
allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_
constants::match_flag_type __flags=regex_constants::match_default)
• template<typename _Ch_traits, typename _Ch_alloc, typename _Allocator, typename _Ch_
type, typename _Rx_traits >
bool regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc
> &__s, match_results< typename basic_string< _Ch_type, _Ch_traits,
_Ch_alloc >::const_iterator, _Allocator > &__m, const basic_regex< _
Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_
constants::match_default)
• template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type
>
_Out_iter regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last,
const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string<
_Ch_type > &__fmt, regex_constants::match_flag_type __flags=regex_
constants::match_default)
• template<typename _Rx_traits, typename _Ch_type >
basic_string< _Ch_type > regex_replace (const basic_string< _Ch_
type > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, const
basic_string< _Ch_type > &__fmt, regex_constants::match_flag_type __
flags=regex_constants::match_default)

```

Variables

- `enable_if`< (!is_member_pointer< _Functor >::value &&!is_function< _Functor >::value &&!is_function< typename remove_pointer< _Functor >::type >::value), typename result_of< _Functor(_Args...)>::type >::type >_invoke (_Functor &__f, _Args &&...__args)
- static ios_base::Init `__ioinit`
- function< void()> `__once_functor`
- constexpr `allocator_arg_t` `allocator_arg`
- const `_Swallow_assign` `ignore`
- `error_code` `make_error_code` (errc)
- `error_condition` `make_error_condition` (errc)
- const `nothrow_t` `nothrow`
- constexpr `piecewise_construct_t` `piecewise_construct`

Standard Stream Objects

The `<iostream>` header declares the eight standard stream objects. For other declarations, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch24.html> and the *I/O forward declarations*

They are required by default to cooperate with the global C library's `FILE` streams, and to be available during program startup and termination. For more information, see the *HOWTO* linked to above.

- [istream cin](#)
- [ostream cout](#)
- [ostream cerr](#)
- [ostream clog](#)
- [wistream wcin](#)
- [wostream wcout](#)
- [wostream wcerr](#)
- [wostream wclog](#)

4.11.1 Detailed Description

ISO C++ entities toplevel namespace is std.

4.11.2 Typedef Documentation

4.11.2.1 typedef void(* std::new_handler)()

If you write your own error handler to be called by `new`, it must be of this type.
Definition at line 75 of file `new`.

4.11.2.2 typedef long long std::streamoff

Type used by `fpos`, [char_traits<char>](#), and [char_traits<wchar_t>](#).

In clauses 21.1.3.1 and 27.4.1 `streamoff` is described as an implementation defined type. Note: In versions of GCC up to and including GCC 3.3, `streamoff` was typedef `long`.

Definition at line 96 of file `postypes.h`.

4.11.2.3 typedef fpos<mbstate_t> std::streampos

File position for char streams.

Definition at line 230 of file postypes.h.

4.11.2.4 typedef ptrdiff_t std::streamsize

Integral type for I/O operation counts and buffer sizes.

Definition at line 100 of file postypes.h.

4.11.2.5 typedef fpos<mbstate_t> std::u16streampos

File position for char16_t streams.

Definition at line 236 of file postypes.h.

4.11.2.6 typedef fpos<mbstate_t> std::u32streampos

File position for char32_t streams.

Definition at line 238 of file postypes.h.

4.11.2.7 typedef fpos<mbstate_t> std::wstreampos

File position for wchar_t streams.

Definition at line 232 of file postypes.h.

4.11.3 Enumeration Type Documentation

4.11.3.1 anonymous enum

Todo

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html This controls some aspect of the sort routines.

Definition at line 2171 of file stl_algo.h.

4.11.3.2 enum std::float_denorm_style

Describes the denormalization for floating-point types.

These values represent the presence or absence of a variable number of exponent bits. This type is used in the [std::numeric_limits](#) class.

Enumerator:

denorm_indeterminate Indeterminate at compile time whether denormalized values are allowed.

denorm_absent The type does not allow denormalized values.

denorm_present The type allows denormalized values.

Definition at line 172 of file limits.

4.11.3.3 enum std::float_round_style

Describes the rounding style for floating-point types.

This is used in the [std::numeric_limits](#) class.

Enumerator:

round_toward_zero Intermediate.

round_to_nearest To zero.

round_toward_infinity To the nearest representable value.

round_toward_neg_infinity To infinity.

Definition at line 157 of file limits.

4.11.4 Function Documentation

4.11.4.1 template<typename _RandomAccessIterator > void std::__final_insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)

This is a helper function for the sort routine.

Definition at line 2176 of file stl_algo.h.

References [__insertion_sort\(\)](#), and [__unguarded_insertion_sort\(\)](#).

Referenced by [sort\(\)](#).

4.11.4.2 `template<typename _RandomAccessIterator, typename _Compare >
void std::__final_insertion_sort (_RandomAccessIterator __first,
_RandomAccessIterator __last, _Compare __comp)`

This is a helper function for the sort routine.

Definition at line 2191 of file `stl_algo.h`.

References `__insertion_sort()`, and `__unguarded_insertion_sort()`.

4.11.4.3 `template<typename _InputIterator, typename _Tp > _InputIterator
std::__find (_InputIterator __first, _InputIterator __last, const _Tp
& __val, input_iterator_tag) [inline]`

This is an overload used by `find()` for the Input Iterator case.

Definition at line 132 of file `stl_algo.h`.

Referenced by `find()`.

4.11.4.4 `template<typename _RandomAccessIterator, typename _Tp >
_RandomAccessIterator std::__find (_RandomAccessIterator
__first, _RandomAccessIterator __last, const _Tp & __val,
random_access_iterator_tag)`

This is an overload used by `find()` for the RAI case.

Definition at line 154 of file `stl_algo.h`.

4.11.4.5 `template<typename _InputIterator, typename _Predicate >
_InputIterator std::__find_if (_InputIterator __first, _InputIterator
__last, _Predicate __pred, input_iterator_tag) [inline]`

This is an overload used by `find_if()` for the Input Iterator case.

Definition at line 143 of file `stl_algo.h`.

Referenced by `find_if()`.

4.11.4.6 `template<typename _RandomAccessIterator, typename _Predicate >
_RandomAccessIterator std::__find_if (_RandomAccessIterator
__first, _RandomAccessIterator __last, _Predicate __pred,
random_access_iterator_tag)`

This is an overload used by `find_if()` for the RAI case.

Definition at line 202 of file `stl_algo.h`.

4.11.4.7 `template<typename _InputIterator, typename _Predicate >
_InputIterator std::__find_if_not (_InputIterator __first,
_InputIterator __last, _Predicate __pred, input_iterator_tag)
[inline]`

This is an overload used by `find_if_not()` for the Input Iterator case.

Definition at line 251 of file `stl_algo.h`.

Referenced by `find_if_not()`.

4.11.4.8 `template<typename _RandomAccessIterator, typename _Predicate >
_RandomAccessIterator std::__find_if_not (_RandomAccessIterator
__first, _RandomAccessIterator __last, _Predicate __pred,
random_access_iterator_tag)`

This is an overload used by `find_if_not()` for the RAI case.

Definition at line 262 of file `stl_algo.h`.

4.11.4.9 `template<typename _EuclideanRingElement >
_EuclideanRingElement std::__gcd (_EuclideanRingElement __m,
_EuclideanRingElement __n)`

This is a helper function for the rotate algorithm specialized on RAIs. It returns the greatest common divisor of two integer values.

Definition at line 1491 of file `stl_algo.h`.

4.11.4.10 `template<typename _RandomAccessIterator > void
std::__heap_select (_RandomAccessIterator __first,
_RandomAccessIterator __middle, _RandomAccessIterator __last)`

This is a helper function for the sort routines.

Definition at line 1899 of file `stl_algo.h`.

References `make_heap()`.

Referenced by `partial_sort()`.

4.11.4.11 `template<typename _RandomAccessIterator , typename _Compare
> void std::__heap_select (_RandomAccessIterator __first,
_RandomAccessIterator __middle, _RandomAccessIterator __last,
_Compare __comp)`

This is a helper function for the sort routines.

Definition at line 1912 of file `stl_algo.h`.

References `make_heap()`.

4.11.4.12 `template<typename _ForwardIterator , typename
_Predicate , typename _Distance > _ForwardIterator
std::__inplace_stable_partition (_ForwardIterator __first,
_ForwardIterator __last, _Predicate __pred, _Distance __len)`

This is a helper function...

Definition at line 1776 of file `stl_algo.h`.

References `advance()`, `distance()`, and `rotate()`.

Referenced by `stable_partition()`.

4.11.4.13 `template<typename _RandomAccessIterator > void
std::__inplace_stable_sort (_RandomAccessIterator __first,
_RandomAccessIterator __last)`

This is a helper function for the stable sorting routines.

Definition at line 3350 of file `stl_algo.h`.

References `__insetion_sort()`, and `__merge_without_buffer()`.

Referenced by `__inplace_stable_sort()`, and `stable_sort()`.

4.11.4.14 `template<typename _RandomAccessIterator, typename _Compare
> void std::__inplace_stable_sort (_RandomAccessIterator __first,
_RandomAccessIterator __last, _Compare __comp)`

This is a helper function for the stable sorting routines.

Definition at line 3369 of file `stl_algo.h`.

References `__inplace_stable_sort()`, `__insetion_sort()`, and `__merge_without_buffer()`.

4.11.4.15 `template<typename _RandomAccessIterator > void
std::__insetion_sort (_RandomAccessIterator __first,
_RandomAccessIterator __last)`

This is a helper function for the sort routine.

Definition at line 2099 of file `stl_algo.h`.

References `__unguarded_linear_insert()`.

Referenced by `__final_insetion_sort()`, and `__inplace_stable_sort()`.

4.11.4.16 `template<typename _RandomAccessIterator, typename _Compare
> void std::__insetion_sort (_RandomAccessIterator __first,
_RandomAccessIterator __last, _Compare __comp)`

This is a helper function for the sort routine.

Definition at line 2122 of file `stl_algo.h`.

References `__unguarded_linear_insert()`.

4.11.4.17 `template<typename _RandomAccessIterator , typename _Size
> void std::__introsort_loop (_RandomAccessIterator __first,
_RandomAccessIterator __last, _Size __depth_limit)`

This is a helper function for the sort routine.

Definition at line 2271 of file `stl_algo.h`.

References `__unguarded_partition_pivot()`.

Referenced by `__introsort_loop()`, and `sort()`.

4.11.4.18 `template<typename _RandomAccessIterator , typename
_Size , typename _Compare > void std::__introsort_loop (
_RandomAccessIterator __first, _RandomAccessIterator __last,
_Size __depth_limit, _Compare __comp)`

This is a helper function for the sort routine.

Definition at line 2293 of file `stl_algo.h`.

References `__introsort_loop()`, and `__unguarded_partition_pivot()`.

4.11.4.19 `template<typename _Size > _Size std::__lg (_Size __n)
[inline]`

This is a helper function for the sort routines and for [random.tcc](#).

Definition at line 972 of file `stl_algbase.h`.

Referenced by `nth_element()`, `std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed()`, and `sort()`.

4.11.4.20 `template<typename _BidirectionalIterator , typename
_Distance , typename _Pointer > void std::__merge_adaptive (
_BidirectionalIterator __first, _BidirectionalIterator __middle,
_BidirectionalIterator __last, _Distance __len1, _Distance __len2,
_Pointer __buffer, _Distance __buffer_size)`

This is a helper function for the merge routines.

Definition at line 2826 of file `stl_algo.h`.

References `__merge_backward()`, `__rotate_adaptive()`, `advance()`, `distance()`, `lower_bound()`, and `upper_bound()`.

Referenced by `__merge_adaptive()`, and `inplace_merge()`.

4.11.4.21 `template<typename _BidirectionalIterator , typename
_Distance , typename _Pointer , typename _Compare >
void std::__merge_adaptive (_BidirectionalIterator __first,
_BidirectionalIterator __middle, _BidirectionalIterator __last,
_Distance __len1, _Distance __len2, _Pointer __buffer, _Distance
__buffer_size, _Compare __comp)`

This is a helper function for the merge routines.

Definition at line 2888 of file `stl_algo.h`.

References `__merge_adaptive()`, `__merge_backward()`, `__rotate_adaptive()`, `advance()`, `distance()`, `lower_bound()`, and `upper_bound()`.

4.11.4.22 `template<typename _BidirectionalIterator1 , typename
_BidirectionalIterator2 , typename _BidirectionalIterator3
> _BidirectionalIterator3 std::__merge_backward (
_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1,
_BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2,
_BidirectionalIterator3 __result)`

This is a helper function for the merge routines.

Definition at line 2723 of file `stl_algo.h`.

Referenced by `__merge_adaptive()`.

4.11.4.23 `template<typename _BidirectionalIterator1 , typename
_BidirectionalIterator2 , typename _BidirectionalIterator3
, typename _Compare > _BidirectionalIterator3
std::__merge_backward (_BidirectionalIterator1 __first1,
_BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2,
_BidirectionalIterator2 __last2, _BidirectionalIterator3 __result,
_Compare __comp)`

This is a helper function for the merge routines.

Definition at line 2758 of file stl_algo.h.

4.11.4.24 `template<typename _BidirectionalIterator , typename _Distance >
void std::__merge_without_buffer (_BidirectionalIterator __first,
_BidirectionalIterator __middle, _BidirectionalIterator __last,
_Distance __len1, _Distance __len2)`

This is a helper function for the merge routines.

Definition at line 2951 of file stl_algo.h.

References `advance()`, `distance()`, `iter_swap()`, `lower_bound()`, `rotate()`, and `upper_bound()`.

Referenced by `__inplace_stable_sort()`, `__merge_without_buffer()`, and `inplace_merge()`.

4.11.4.25 `template<typename _BidirectionalIterator , typename _Distance
, typename _Compare > void std::__merge_without_buffer (
_BidirectionalIterator __first, _BidirectionalIterator __middle,
_BidirectionalIterator __last, _Distance __len1, _Distance __len2,
_Compare __comp)`

This is a helper function for the merge routines.

Definition at line 2995 of file stl_algo.h.

References `__merge_without_buffer()`, `advance()`, `distance()`, `iter_swap()`, `lower_bound()`, `rotate()`, and `upper_bound()`.

4.11.4.26 `template<typename _Iterator > void std::__move_median_first (
_Iterator __a, _Iterator __b, _Iterator __c)`

Swaps the median value of `*__a`, `*__b` and `*__c` to `*__a`.

Definition at line 80 of file stl_algo.h.

References `iter_swap()`.

Referenced by `__unguarded_partition_pivot()`.

4.11.4.27 `template<typename _Iterator , typename _Compare > void
std::__move_median_first (_Iterator __a, _Iterator __b, _Iterator
__c, _Compare __comp)`

Swaps the median value of *__a, *__b and *__c under __comp to *__a.

Definition at line 104 of file stl_algo.h.

References iter_swap().

4.11.4.28 `template<typename _ForwardIterator , typename _Predicate >
_ForwardIterator std::__partition (_ForwardIterator __first,
_ForwardIterator __last, _Predicate __pred, forward_iterator_tag
)`

This is a helper function...

Definition at line 1721 of file stl_algo.h.

References iter_swap().

Referenced by partition().

4.11.4.29 `template<typename _BidirectionalIterator , typename _Predicate
> _BidirectionalIterator std::__partition (_BidirectionalIterator
__first, _BidirectionalIterator __last, _Predicate __pred,
bidirectional_iterator_tag)`

This is a helper function...

Definition at line 1746 of file stl_algo.h.

References iter_swap().

4.11.4.30 `template<typename _BidirectionalIterator > void std::__reverse
(_BidirectionalIterator __first, _BidirectionalIterator __last,
bidirectional_iterator_tag)`

This is an uglified reverse(_BidirectionalIterator, _BidirectionalIterator) overloaded for bidirectional iterators.

Definition at line 1391 of file stl_algo.h.

References iter_swap().

Referenced by `__rotate()`, and `reverse()`.

4.11.4.31 `template<typename _RandomAccessIterator > void std::__reverse (`
`_RandomAccessIterator __first, _RandomAccessIterator __last,`
`random_access_iterator_tag)`

This is an uglified `reverse(_BidirectionalIterator, _BidirectionalIterator)` overloaded for random access iterators.

Definition at line 1411 of file `stl_algo.h`.

References `iter_swap()`.

4.11.4.32 `template<typename _ForwardIterator > void std::__rotate`
`(_ForwardIterator __first, _ForwardIterator __middle,`
`_FowardIterator __last, forward_iterator_tag)`

This is a helper function for the rotate algorithm.

Definition at line 1505 of file `stl_algo.h`.

References `iter_swap()`.

Referenced by `__gnu_cxx::bitmap_allocator< _Tp >::_M_deallocate_single_object()`, and `rotate()`.

4.11.4.33 `template<typename _BidirectionalIterator > void std::__rotate (`
`_BidirectionalIterator __first, _BidirectionalIterator __middle,`
`_BidirectionalIterator __last, bidirectional_iterator_tag)`

This is a helper function for the rotate algorithm.

Definition at line 1541 of file `stl_algo.h`.

References `__reverse()`, and `iter_swap()`.

4.11.4.34 `template<typename _RandomAccessIterator > void std::__rotate (`
`_RandomAccessIterator __first, _RandomAccessIterator __middle,`
`_RandomAccessIterator __last, random_access_iterator_tag)`

This is a helper function for the rotate algorithm.

Definition at line 1571 of file `stl_algo.h`.

References `iter_swap()`, and `swap_ranges()`.

4.11.4.35 `template<typename _BidirectionalIterator1 , typename
_BidirectionalIterator2 , typename _Distance >
_BidirectionalIterator1 std::__rotate_adaptive (
_BidirectionalIterator1 __first, _BidirectionalIterator1 __middle,
_BidirectionalIterator1 __last, _Distance __len1, _Distance __len2,
_BidirectionalIterator2 __buffer, _Distance __buffer_size)`

This is a helper function for the merge routines.

Definition at line 2794 of file `stl_algo.h`.

References `advance()`, `distance()`, and `rotate()`.

Referenced by `__merge_adaptive()`.

4.11.4.36 `template<typename _ForwardIterator , typename _Integer ,
typename _Tp , typename _BinaryPredicate > _ForwardIterator
std::__search_n (_ForwardIterator __first, _ForwardIterator
__last, _Integer __count, const _Tp & __val, _BinaryPredicate
__binary_pred, std::forward_iterator_tag)`

This is an uglified `search_n(_ForwardIterator, _ForwardIterator, _Integer, const
_Tp&, _BinaryPredicate)` overloaded for forward iterators.

Definition at line 414 of file `stl_algo.h`.

4.11.4.37 `template<typename _RandomAccessIter , typename _Integer ,
typename _Tp , typename _BinaryPredicate > _RandomAccessIter
std::__search_n (_RandomAccessIter __first, _RandomAccessIter
__last, _Integer __count, const _Tp & __val, _BinaryPredicate
__binary_pred, std::random_access_iterator_tag)`

This is an uglified `search_n(_ForwardIterator, _ForwardIterator, _Integer, const
_Tp&, _BinaryPredicate)` overloaded for random access iterators.

Definition at line 453 of file `stl_algo.h`.

4.11.4.38 `template<typename _RandomAccessIter , typename _Integer
, typename _Tp > _RandomAccessIter std::__search_n (`
`_RandomAccessIter __first, _RandomAccessIter __last, _Integer`
`__count, const _Tp & __val, std::random_access_iterator_tag)`

This is an uglified `search_n(_ForwardIterator, _ForwardIterator, _Integer, const _Tp&)` overloaded for random access iterators.

Definition at line 360 of file `stl_algo.h`.

4.11.4.39 `template<typename _ForwardIterator , typename _Integer
, typename _Tp > _ForwardIterator std::__search_n (`
`_ForwardIterator __first, _ForwardIterator __last, _Integer`
`__count, const _Tp & __val, std::forward_iterator_tag)`

This is an uglified `search_n(_ForwardIterator, _ForwardIterator, _Integer, const _Tp&)` overloaded for forward iterators.

Definition at line 328 of file `stl_algo.h`.

Referenced by `search_n()`.

4.11.4.40 `template<typename _ForwardIterator , typename _Pointer ,
typename _Predicate , typename _Distance > _ForwardIterator`
`std::__stable_partition_adaptive (_ForwardIterator __first,`
`_ForwardIterator __last, _Predicate __pred, _Distance __len,`
`_Pointer __buffer, _Distance __buffer_size)`

This is a helper function...

Definition at line 1801 of file `stl_algo.h`.

References `advance()`, `distance()`, and `rotate()`.

Referenced by `stable_partition()`.

4.11.4.41 `template<typename _RandomAccessIterator > void`
`std::__unguarded_insertion_sort (_RandomAccessIterator __first,`
`_RandomAccessIterator __last) [inline]`

This is a helper function for the sort routine.

Definition at line 2144 of file `stl_algo.h`.

References `__unguarded_linear_insert()`.

Referenced by `__final_insertion_sort()`.

4.11.4.42 `template<typename _RandomAccessIterator, typename _Compare
> void std::__unguarded_insertion_sort (_RandomAccessIterator
__first, _RandomAccessIterator __last, _Compare __comp)
[inline]`

This is a helper function for the sort routine.

Definition at line 2157 of file `stl_algo.h`.

References `__unguarded_linear_insert()`.

4.11.4.43 `template<typename _RandomAccessIterator, typename _Compare
> void std::__unguarded_linear_insert (_RandomAccessIterator
__last, _Compare __comp)`

This is a helper function for the sort routine.

Definition at line 2080 of file `stl_algo.h`.

4.11.4.44 `template<typename _RandomAccessIterator > void
std::__unguarded_linear_insert (_RandomAccessIterator __last)`

This is a helper function for the sort routine.

Definition at line 2062 of file `stl_algo.h`.

Referenced by `__insertion_sort()`, and `__unguarded_insertion_sort()`.

4.11.4.45 `template<typename _RandomAccessIterator, typename
_Tp > _RandomAccessIterator std::__unguarded_partition (
_RandomAccessIterator __first, _RandomAccessIterator __last,
const _Tp & __pivot)`

This is a helper function...

Definition at line 2207 of file `stl_algo.h`.

References `iter_swap()`.

Referenced by `__unguarded_partition_pivot()`.

4.11.4.46 `template<typename _RandomAccessIterator, typename
_Tp, typename _Compare> _RandomAccessIterator
std::__unguarded_partition (_RandomAccessIterator __first,
_RandomAccessIterator __last, const _Tp & __pivot, _Compare
__comp)`

This is a helper function...

Definition at line 2227 of file `stl_algo.h`.

References `iter_swap()`.

4.11.4.47 `template<typename _RandomAccessIterator >
_RandomAccessIterator std::__unguarded_partition_pivot (
_RandomAccessIterator __first, _RandomAccessIterator __last)
[inline]`

This is a helper function...

Definition at line 2248 of file `stl_algo.h`.

References `__move_median_first()`, and `__unguarded_partition()`.

Referenced by `__introsort_loop()`.

4.11.4.48 `template<typename _RandomAccessIterator, typename _Compare
> _RandomAccessIterator std::__unguarded_partition_pivot (
_RandomAccessIterator __first, _RandomAccessIterator __last,
_Compare __comp) [inline]`

This is a helper function...

Definition at line 2260 of file `stl_algo.h`.

References `__move_median_first()`, and `__unguarded_partition()`.

4.11.4.49 `template<typename _ForwardIterator, typename _OutputIterator
> _OutputIterator std::__unique_copy (_ForwardIterator
__first, _ForwardIterator __last, _OutputIterator __result,
forward_iterator_tag, output_iterator_tag)`

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator)` overloaded for forward iterators and output iterator as result.

Definition at line 1243 of file `stl_algo.h`.

Referenced by `unique_copy()`.

4.11.4.50 `template<typename _InputIterator, typename _OutputIterator
> _OutputIterator std::__unique_copy (_InputIterator __first,
_InputIterator __last, _OutputIterator __result, input_iterator_tag
, output_iterator_tag)`

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator)` overloaded for input iterators and output iterator as result.

Definition at line 1266 of file `stl_algo.h`.

4.11.4.51 `template<typename _InputIterator, typename _ForwardIterator
> _ForwardIterator std::__unique_copy (_InputIterator
__first, _InputIterator __last, _ForwardIterator __result,
input_iterator_tag, forward_iterator_tag)`

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator)` overloaded for input iterators and forward iterator as result.

Definition at line 1289 of file `stl_algo.h`.

4.11.4.52 `template<typename _ForwardIterator, typename _OutputIterator ,
typename _BinaryPredicate > _OutputIterator std::__unique_copy (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator
__result, _BinaryPredicate __binary_pred, forward_iterator_tag ,
output_iterator_tag)`

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator, _BinaryPredicate)` overloaded for forward iterators and output iterator as result.

Definition at line 1310 of file `stl_algo.h`.

4.11.4.53 `template<typename _InputIterator, typename _OutputIterator,
typename _BinaryPredicate > _OutputIterator std::__unique_copy (
 _InputIterator __first, _InputIterator __last, _OutputIterator
 __result, _BinaryPredicate __binary_pred, input_iterator_tag,
 output_iterator_tag)`

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator, _BinaryPredicate)` overloaded for input iterators and output iterator as result.

Definition at line 1339 of file `stl_algo.h`.

4.11.4.54 `template<typename _InputIterator, typename _ForwardIterator,
typename _BinaryPredicate > _ForwardIterator std::__unique_copy
(_InputIterator __first, _InputIterator __last, _ForwardIterator
 __result, _BinaryPredicate __binary_pred, input_iterator_tag,
 forward_iterator_tag)`

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator, _BinaryPredicate)` overloaded for input iterators and forward iterator as result.

Definition at line 1368 of file `stl_algo.h`.

4.11.4.55 `template<typename _T1, typename... _Args> void std::_Construct (
 _T1 * __p, _Args &&... __args) [inline]`

Constructs an object in existing memory by invoking an allocated object's constructor with an initializer.

Definition at line 75 of file `stl_construct.h`.

4.11.4.56 `template<typename _Tp > void std::_Destroy (_Tp * __pointer)
[inline]`

Destroy the object pointed to by a pointer type.

Definition at line 93 of file `stl_construct.h`.

Referenced by `std::deque< _Tp, _Alloc >::_M_fill_initialize()`, `std::deque< _Tp, _Alloc >::_M_range_initialize()`, `std::vector< _Tp, _Alloc >::operator=()`, `std::vector< _Tp, _Alloc >::reserve()`, and `std::vector< std::sub_match< _Bi_iter >, _Allocator >::~~vector()`.

4.11.4.57 `template<typename _ForwardIterator > void std::_Destroy (`
`_FowardIterator __first, _ForwardIterator __last) [inline]`

Destroy a range of objects. If the value_type of the object has a trivial destructor, the compiler should optimize all of this away, otherwise the objects' destructors must be invoked.

Definition at line 123 of file stl_construct.h.

4.11.4.58 `template<typename _InputIterator , typename _Tp > _Tp`
`std::accumulate (_InputIterator __first, _InputIterator __last, _Tp`
`__init) [inline]`

Accumulate values in a range.

Accumulates the values in the range [first,last) using operator+(). The initial value is *init*. The values are processed in order.

Parameters

first Start of range.

last End of range.

init Starting value to add other values to.

Returns

The final sum.

Definition at line 121 of file stl_numeric.h.

4.11.4.59 `template<typename _InputIterator , typename _Tp , typename`
`_BinaryOperation > _Tp std::accumulate (_InputIterator __first,`
`_InputIterator __last, _Tp __init, _BinaryOperation __binary_op)`
`[inline]`

Accumulate values in a range with operation.

Accumulates the values in the range [first,last) using the function object *binary_op*. The initial value is *init*. The values are processed in order.

Parameters

first Start of range.

last End of range.

init Starting value to add other values to.

binary_op Function object to accumulate with.

Returns

The final sum.

Definition at line 147 of file `std_numeric.h`.

4.11.4.60 `template<typename _Tp> std::complex<_Tp> std::acos (const
std::complex<_Tp> & __z) [inline]`

`acos(__z)` [8.1.2].

Definition at line 1580 of file `complex`.

4.11.4.61 `template<typename _Tp> std::complex<_Tp> std::acosh (const
std::complex<_Tp> & __z) [inline]`

`acosh(__z)` [8.1.5].

Definition at line 1699 of file `complex`.

4.11.4.62 `template<typename _Tp> _Tp* std::addressof (_Tp & __r)
[inline]`

`declval`, from `type_traits`.

Returns the actual address of the object or function referenced by `r`, even in the presence of an overloaded operator`&`.

Parameters

`__r` Reference to an object or function.

Returns

The actual address.

Definition at line 96 of file `move.h`.

4.11.4.63 `template<typename _InputIterator, typename _OutputIterator >
_OutputIterator std::adjacent_difference (_InputIterator __first,
_InputIterator __last, _OutputIterator __result)`

Return differences between adjacent values.

Computes the difference between adjacent values in the range [first,last) using operator-() and writes the result to *result*.

Parameters

first Start of input range.

last End of input range.

result Output to write sums to.

Returns

Iterator pointing just beyond the values written to result.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 539. `partial_sum` and `adjacent_difference` should mention requirements

Definition at line 317 of file `std_numeric.h`.

4.11.4.64 `template<typename _InputIterator, typename _OutputIterator
, typename _BinaryOperation > _OutputIterator
std::adjacent_difference (_InputIterator __first, _InputIterator
__last, _OutputIterator __result, _BinaryOperation __binary_op)`

Return differences between adjacent values.

Computes the difference between adjacent values in the range [first,last) using the function object *binary_op* and writes the result to *result*.

Parameters

first Start of input range.

last End of input range.

result Output to write sums to.

Returns

Iterator pointing just beyond the values written to result.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 539. `partial_sum` and `adjacent_difference` should mention requirements

Definition at line 359 of file `stl_numeric.h`.

4.11.4.65 `template<typename _InputIterator, typename _Distance> void
std::advance(_InputIterator & __i, _Distance __n) [inline]`

A generalization of pointer arithmetic.

Parameters

- i* An input iterator.
- n* The *delta* by which to change *i*.

Returns

Nothing.

This increments *i* by *n*. For bidirectional and random access iterators, *n* may be negative, in which case *i* is decremented.

For random access iterators, this uses their `+` and `-` operations and are constant time. For other iterator classes they are linear time.

Definition at line 171 of file `stl_iterator_base_funcs.h`.

References `__iterator_category()`.

Referenced by `__inplace_stable_partition()`, `__merge_adaptive()`, `__merge_without_buffer()`, `__rotate_adaptive()`, `__stable_partition_adaptive()`, `std::deque< _Tp, _Alloc >::_M_range_initialize()`, `equal_range()`, `is_permutation()`, `lower_bound()`, `partition_point()`, and `upper_bound()`.

4.11.4.66 `template<typename _Tp> __gnu_cxx::__promote<_Tp>::__type
std::arg(_Tp __x) [inline]`

Additional overloads [8.1.9].

Definition at line 1797 of file `complex`.

References `arg()`.

4.11.4.67 `template<typename _Tp> std::complex<_Tp> std::asin (const
std::complex<_Tp> & __z) [inline]`

`asin(__z)` [8.1.3].

Definition at line 1616 of file `complex`.

4.11.4.68 `template<typename _Tp> std::complex<_Tp> std::asinh (const
std::complex<_Tp> & __z) [inline]`

`asinh(__z)` [8.1.6].

Definition at line 1738 of file `complex`.

4.11.4.69 `template<typename _Tp> std::complex<_Tp> std::atan (const
std::complex<_Tp> & __z) [inline]`

`atan(__z)` [8.1.4].

Definition at line 1660 of file `complex`.

4.11.4.70 `template<typename _Tp> std::complex<_Tp> std::atanh (const
std::complex<_Tp> & __z) [inline]`

`atanh(__z)` [8.1.7].

Definition at line 1782 of file `complex`.

4.11.4.71 `template<class _Tp> constexpr const _Tp* std::begin (
initializer_list<_Tp> __ils)`

Return an iterator pointing to the first element of the `initilizer_list`.

Parameters

il Initializer list.

Definition at line 86 of file `initializer_list`.

Referenced by `std::list<_Tp, _Alloc>::merge()`.

4.11.4.72 `template<class _Container > auto std::begin (_Container & __cont) [inline]`

Return an iterator pointing to the first element of the container.

Parameters

cont Container.

Definition at line 48 of file `range_access.h`.

4.11.4.73 `template<class _Container > auto std::begin (const _Container & __cont) [inline]`

Return an iterator pointing to the first element of the const container.

Parameters

cont Container.

Definition at line 58 of file `range_access.h`.

4.11.4.74 `template<class _Tp , size_t _Nm> _Tp* std::begin (_Tp(& __arr[_Nm]) [inline]`

Return an iterator pointing to the first element of the array.

Parameters

arr Array.

Definition at line 87 of file `range_access.h`.

4.11.4.75 `template<typename _Result , typename _Functor ,
typename... _ArgTypes> _Bindres_helper<_Result, _Functor,
_ArgTypes...>::type std::bind (_Functor && __f, _ArgTypes &&...
__args) [inline]`

Function template for `std::bind<R>`.

Definition at line 1462 of file `functional`.

4.11.4.76 `ios_base& std::boolalpha (ios_base & __base) [inline]`

Calls `base.setf(ios_base::boolalpha)`.

Definition at line 797 of file `ios_base.h`.

References `std::ios_base::boolalpha`.

4.11.4.77 `template<typename _Tp , typename _Tp1 , _Lock_policy _Lp> __shared_ptr<_Tp, _Lp> std::const_pointer_cast (const __shared_ptr<_Tp1, _Lp> & __r) [inline]`

`const_pointer_cast`

Definition at line 1129 of file `shared_ptr_base.h`.

4.11.4.78 `template<typename _Tp > reference_wrapper<const _Tp> std::cref (const _Tp & __t) [inline]`

Denotes a const reference should be taken to a variable.

Definition at line 476 of file `functional`.

Referenced by `cref()`.

4.11.4.79 `template<typename _Tp > reference_wrapper<const _Tp> std::cref (reference_wrapper<_Tp> __t) [inline]`

Partial specialization.

Definition at line 488 of file `functional`.

References `cref()`.

4.11.4.80 `ios_base& std::dec (ios_base & __base) [inline]`

Calls `base.setf(ios_base::dec, ios_base::basefield)`.

Definition at line 935 of file `ios_base.h`.

References `std::ios_base::basefield`, `std::ios_base::dec`, and `std::ios_base::setf()`.

Referenced by `operator>>()`.

4.11.4.81 `template<typename _InputIterator > iterator_traits<_
_InputIterator>::difference_type std::distance (_InputIterator
__first, _InputIterator __last) [inline]`

A generalization of pointer arithmetic.

Parameters

first An input iterator.

last An input iterator.

Returns

The distance between them.

Returns `n` such that `first + n == last`. This requires that `last` must be reachable from `first`. Note that `n` may be negative.

For random access iterators, this uses their `+` and `-` operations and are constant time. For other iterator classes they are linear time.

Definition at line 113 of file `stl_iterator_base_funcs.h`.

References `__iterator_category()`.

Referenced by `__inplace_stable_partition()`, `__merge_adaptive()`, `__merge_without_buffer()`, `__rotate_adaptive()`, `__stable_partition_adaptive()`, `std::deque<_Tp, _Alloc>::M_range_initialize()`, `equal_range()`, `inplace_merge()`, `is_heap_until()`, `is_permutation()`, `std::sub_match<_Bi_iter>::length()`, `lower_bound()`, `__gnu_parallel::multiseq_partition()`, `__gnu_parallel::multiseq_selection()`, `partition_point()`, `std::match_results<_FwdIterT, _Alloc>::position()`, `std::list<_Tp, _Alloc>::size()`, and `upper_bound()`.

4.11.4.82 `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>
__shared_ptr<_Tp, _Lp> std::dynamic_pointer_cast (const
__shared_ptr<_Tp1, _Lp> & __r) [inline]`

`dynamic_pointer_cast`

Definition at line 1139 of file shared_ptr_base.h.

4.11.4.83 `template<class _Tp > constexpr const _Tp* std::end (`
`initializer_list< _Tp > __ils)`

Return an iterator pointing to one past the last element of the initializer_list.

Parameters

il Initializer list.

Definition at line 96 of file initializer_list.

Referenced by std::list< _Tp, _Alloc >::merge().

4.11.4.84 `template<class _Container > auto std::end (_Container & __cont)`
`[inline]`

Return an iterator pointing to one past the last element of the container.

Parameters

cont Container.

Definition at line 68 of file range_access.h.

4.11.4.85 `template<class _Container > auto std::end (const _Container &`
`__cont) [inline]`

Return an iterator pointing to one past the last element of the const container.

Parameters

cont Container.

Definition at line 78 of file range_access.h.

4.11.4.86 `template<class _Tp, size_t _Nm> _Tp* std::end (_Tp(&)
__arr[_Nm]) [inline]`

Return an iterator pointing to one past the last element of the array.

Parameters

arr Array.

Definition at line 97 of file range_access.h.

4.11.4.87 `template<typename _CharT, typename _Traits >
basic_ostream<_CharT, _Traits>& std::endl (basic_ostream<
_CharT, _Traits > & __os) [inline]`

Write a newline and flush the stream.

This manipulator is often mistakenly used when a simple new-line is desired, leading to poor buffering performance. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more on this subject.

Definition at line 543 of file ostream.

References flush(), std::basic_ostream< _CharT, _Traits >::put(), and std::basic_ios< _CharT, _Traits >::widen().

4.11.4.88 `template<typename _CharT, typename _Traits >
basic_ostream<_CharT, _Traits>& std::ends (basic_ostream<
_CharT, _Traits > & __os) [inline]`

Write a null character into the output sequence.

Null character is CharT() by definition. For CharT of char, this correctly writes the ASCII NUL character string terminator.

Definition at line 554 of file ostream.

References std::basic_ostream< _CharT, _Traits >::put().

4.11.4.89 `template<typename _Tp> _Tp std::fabs (const std::complex< _Tp
> & __z) [inline]`

`fabs(__z)` [8.1.8].

Definition at line 1791 of file `complex`.

References `abs()`.

4.11.4.90 `ios_base& std::fixed (ios_base & __base) [inline]`

Calls `base.setf(ios_base::fixed, ios_base::floatfield)`.

Definition at line 960 of file `ios_base.h`.

References `std::ios_base::fixed`, `std::ios_base::floatfield`, and `std::ios_base::setf()`.

4.11.4.91 `template<typename _CharT , typename _Traits >
basic_ostream<_CharT, _Traits>& std::flush (basic_ostream<
_CharT, _Traits > & __os) [inline]`

Flushes the output stream.

This manipulator simply calls the stream's `flush()` member function.

Definition at line 564 of file `ostream`.

References `std::basic_ostream< _CharT, _Traits >::flush()`.

Referenced by `endl()`.

4.11.4.92 `template<typename _Tp> _Tp&& std::forward (typename
std::remove_reference< _Tp >::type & __t) [inline]`

`forward` (as per N3143)

Definition at line 62 of file `move.h`.

4.11.4.93 `template<typename _MoneyT > _Get_money<_MoneyT>
std::get_money (_MoneyT & __mon, bool __intl = false)
[inline]`

Extended manipulator for extracting money.

Parameters

mon Either long double or a specialization of `basic_string`.

intl A bool indicating whether international format is to be used.

Sent to a stream object, this manipulator extracts *mon*.

Definition at line 258 of file `iomanip`.

4.11.4.94 `template<typename _Tp > pair<_Tp*, ptrdiff_t>
std::get_temporary_buffer (ptrdiff_t __len)`

Allocates a temporary buffer.

Parameters

len The number of objects of type `Tp`.

Returns

See full description.

Reinventing the wheel, but this time with prettier spokes!

This function tries to obtain storage for `len` adjacent `Tp` objects. The objects themselves are not constructed, of course. A `pair<>` is returned containing *the buffer's address and capacity (in the units of `sizeof(Tp)`)*, or a pair of 0 values if no storage can be obtained. Note that the capacity obtained may be less than that requested if the memory is unavailable; you should compare `len` with the `.second` return value.

Provides the nothrow exception guarantee.

Definition at line 86 of file `stl_tempbuf.h`.

Referenced by `std::_Temporary_buffer<_ForwardIterator, _Tp >::_Temporary_buffer()`.

4.11.4.95 `template<typename _CharT, typename _Traits, typename _Alloc >
 basic_istream< _CharT, _Traits > & std::getline (basic_istream<
 _CharT, _Traits > & __is, basic_string< _CharT, _Traits, _Alloc >
 & __str, _CharT __delim)`

Read a line from stream into a string.

Parameters

is Input stream.

str Buffer to store into.

delim Character marking end of line.

Returns

Reference to the input stream.

Stores characters from *is* into *str* until *delim* is found, the end of the stream is encountered, or *str.max_size()* is reached. If *is.width()* is non-zero, that is the limit on the number of characters stored into *str*. Any previous contents of *str* are erased. If *delim* was encountered, it is extracted but not stored into *str*.

Definition at line 1070 of file `basic_string.tcc`.

References `std::basic_string< _CharT, _Traits, _Alloc >::erase()`, `std::basic_string< _CharT, _Traits, _Alloc >::max_size()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

Referenced by `getline()`.

4.11.4.96 `template<typename _CharT, typename _Traits, typename _Alloc >
 basic_istream< _CharT, _Traits>& std::getline (basic_istream<
 _CharT, _Traits > & __is, basic_string< _CharT, _Traits, _Alloc >
 & __str) [inline]`

Read a line from stream into a string.

Parameters

is Input stream.

str Buffer to store into.

Returns

Reference to the input stream.

Stores characters from *is* into *str* until ‘

’ is found, the end of the stream is encountered, or *str.max_size()* is reached. If *is.width()* is non-zero, that is the limit on the number of characters stored into *str*. Any previous contents of *str* are erased. If end of line was encountered, it is extracted but not stored into *str*.

Definition at line 2734 of file *basic_string.h*.

References *getline()*, and *std::basic_ios< _CharT, _Traits >::widen()*.

4.11.4.97 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > & __is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & __str, _CharT __delim)`

Read a line from stream into a string.

Parameters

- `__is` Input stream.
- `__str` Buffer to store into.
- `__delim` Character marking end of line.

Returns

Reference to the input stream.

Stores characters from `__is` into `__str` until `__delim` is found, the end of the stream is encountered, or *str.max_size()* is reached. If *is.width()* is non-zero, that is the limit on the number of characters stored into `__str`. Any previous contents of `__str` are erased. If *delim* was encountered, it is extracted but not stored into `__str`.

Definition at line 627 of file *vstring.tcc*.

References *__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append()*, *__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::erase()*, *__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::max_size()*, *std::basic_ios< _CharT, _Traits >::rdbuf()*, and *std::basic_ios< _CharT, _Traits >::setstate()*.

4.11.4.98 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> basic_istream<_CharT, _Traits>& std::getline (basic_istream<_CharT, _Traits> & __is, __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base> & __str) [inline]`

Read a line from stream into a string.

Parameters

__is Input stream.

__str Buffer to store into.

Returns

Reference to the input stream.

Stores characters from *is* into *__str* until ' '

' ' is found, the end of the stream is encountered, or *str.max_size()* is reached. If *is.width()* is non-zero, that is the limit on the number of characters stored into *__str*. Any previous contents of *__str* are erased. If end of line was encountered, it is extracted but not stored into *__str*.

Definition at line 2509 of file *vstring.h*.

References *getline()*, and *std::basic_ios<_CharT, _Traits>::widen()*.

4.11.4.99 `template<typename _Facet> bool std::has_facet (const locale & __loc) throw ()`

Test for the presence of a facet.

has_facet tests the *locale* argument for the presence of the facet type provided as the template parameter. Facets derived from the facet parameter will also return true.

Parameters

Facet The facet type to test the presence of.

locale The locale to test.

Returns

true if *locale* contains a facet of type *Facet*, else false.

Definition at line 93 of file *locale_classes.tcc*.

4.11.4.100 ios_base& std::hex (ios_base & __base) [inline]

Calls base.setf(ios_base::hex, ios_base::basefield).

Definition at line 943 of file ios_base.h.

References std::ios_base::basefield, std::ios_base::hex, and std::ios_base::setf().

**4.11.4.101 template<typename _InputIterator1 , typename _InputIterator2 ,
 typename _Tp > _Tp std::inner_product (_InputIterator1 __first1,
 _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init)
 [inline]**

Compute inner product of two ranges.

Starting with an initial value of *init*, multiplies successive elements from the two ranges and adds each product into the accumulated value using operator+(). The values in the ranges are processed in order.

Parameters

first1 Start of range 1.

last1 End of range 1.

first2 Start of range 2.

init Starting value to add other values to.

Returns

The final inner product.

Definition at line 175 of file stl_numeric.h.

**4.11.4.102 template<typename _InputIterator1 , typename _InputIterator2
 , typename _Tp , typename _BinaryOperation1 , typename
 _BinaryOperation2 > _Tp std::inner_product (_InputIterator1
 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp
 __init, _BinaryOperation1 __binary_op1, _BinaryOperation2
 __binary_op2) [inline]**

Compute inner product of two ranges.

Starting with an initial value of *init*, applies *binary_op2* to successive elements from the two ranges and accumulates each result into the accumulated value using *binary_op1*. The values in the ranges are processed in order.

Parameters

first1 Start of range 1.
last1 End of range 1.
first2 Start of range 2.
init Starting value to add other values to.
binary_op1 Function object to accumulate with.
binary_op2 Function object to apply to pairs of input values.

Returns

The final inner product.

Definition at line 207 of file `std_numeric.h`.

4.11.4.103 `ios_base& std::internal (ios_base & __base) [inline]`

Calls `base.setf(ios_base::internal, ios_base::adjustfield)`.

Definition at line 910 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::adjustfield`, and `std::ios_base::internal`.

4.11.4.104 `template<typename _ForwardIterator, typename _Tp> void std::iota (_ForwardIterator __first, _ForwardIterator __last, _Tp __value)`

Create a range of sequentially increasing values.

For each element in the range `[first,last)` assigns `value` and increments `value` as if by `++value`.

Parameters

first Start of range.
last End of range.
value Starting value.

Returns

Nothing.

Definition at line 83 of file `std_numeric.h`.

4.11.4.105 `template<typename _CharT > bool std::isalnum (_CharT __c,
const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::alnum, __c)`.

4.11.4.106 `template<typename _CharT > bool std::isalpha (_CharT __c,
const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::alpha, __c)`.

4.11.4.107 `template<typename _CharT > bool std::isctrl (_CharT __c,
const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::cntrl, __c)`.

4.11.4.108 `template<typename _CharT > bool std::isdigit (_CharT __c,
const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::digit, __c)`.

4.11.4.109 `template<typename _CharT > bool std::isgraph (_CharT __c,
const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::graph, __c)`.

4.11.4.110 `template<typename _CharT > bool std::islower (_CharT __c,
const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::lower, __c)`.

4.11.4.111 `template<typename _CharT > bool std::isprint (_CharT __c,
const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::print, __c)`.

4.11.4.112 `template<typename _CharT > bool std::ispunct (_CharT __c,
const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::punct, __c)`.

4.11.4.113 `template<typename _CharT > bool std::isspace (_CharT __c,
const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::space, __c)`.

4.11.4.114 `template<typename _CharT > bool std::isupper (_CharT __c,
const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::upper, __c)`.

4.11.4.115 `template<typename _CharT > bool std::isxdigit (_CharT __c,
const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::xdigit, __c)`.

4.11.4.116 `ios_base& std::left (ios_base & __base) [inline]`

Calls `base.setf(ios_base::left, ios_base::adjustfield)`.

Definition at line 918 of file `ios_base.h`.

References `std::ios_base::adjustfield`, `std::ios_base::left`, and `std::ios_base::setf()`.

Referenced by operator<<().

4.11.4.117 `template<class _T1 , class _T2 > pair<typename
__decay_and_strip<_T1>::__type, typename
__decay_and_strip<_T2>::__type> std::make_pair (_T1 && __x,
_T2 && __y) [inline]`

A convenience wrapper for creating a pair from two objects.

Parameters

- x* The first object.
- y* The second object.

Returns

A newly-constructed pair<> object of the appropriate type.

The standard requires that the objects be passed by reference-to-const, but LWG issue #181 says they should be passed by const value. We follow the LWG by default.

Definition at line 262 of file `std_pair.h`.

Referenced by `__gnu_parallel::__find_template()`, `__gnu_parallel::__parallel_merge_advance()`, `__gnu_parallel::__parallel_sort_qsb()`, `__gnu_parallel::__qsb_local_sort_with_helping()`, `__gnu_parallel::__find_first_of_selector<_FIterator>::__M_sequential_algorithm()`, `__gnu_parallel::__adjacent_find_selector::__M_sequential_algorithm()`, `__gnu_parallel::__find_if_selector::__M_sequential_algorithm()`, `minmax_element()`, `__gnu_parallel::multiseq_partition()`, `__gnu_parallel::multiseq_selection()`, `__gnu_parallel::parallel_multiway_merge()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

4.11.4.118 `ios_base& std::noboolalpha (ios_base & __base) [inline]`

Calls `base.unsetf(ios_base::boolalpha)`.

Definition at line 805 of file `ios_base.h`.

References `std::ios_base::boolalpha`, and `std::ios_base::unsetf()`.

4.11.4.119 `ios_base& std::noshowbase (ios_base & __base) [inline]`

Calls `base.unsetf(ios_base::showbase)`.

Definition at line 821 of file `ios_base.h`.

References `std::ios_base::showbase`, and `std::ios_base::unsetf()`.

4.11.4.120 ios_base& std::noshowpoint (ios_base & __base) [inline]

Calls `base.unsetf(ios_base::showpoint)`.

Definition at line 837 of file `ios_base.h`.

References `std::ios_base::showpoint`, and `std::ios_base::unsetf()`.

4.11.4.121 ios_base& std::noshowpos (ios_base & __base) [inline]

Calls `base.unsetf(ios_base::showpos)`.

Definition at line 853 of file `ios_base.h`.

References `std::ios_base::showpos`, and `std::ios_base::unsetf()`.

4.11.4.122 ios_base& std::noskipws (ios_base & __base) [inline]

Calls `base.unsetf(ios_base::skipws)`.

Definition at line 869 of file `ios_base.h`.

References `std::ios_base::skipws`, and `std::ios_base::unsetf()`.

4.11.4.123 ios_base& std::nounitbuf (ios_base & __base) [inline]

Calls `base.unsetf(ios_base::unitbuf)`.

Definition at line 901 of file `ios_base.h`.

References `std::ios_base::unitbuf`, and `std::ios_base::unsetf()`.

4.11.4.124 ios_base& std::nouppercase (ios_base & __base) [inline]

Calls `base.unsetf(ios_base::uppercase)`.

Definition at line 885 of file `ios_base.h`.

References `std::ios_base::unsetf()`, and `std::ios_base::uppercase`.

4.11.4.125 `ios_base& std::oct (ios_base & __base) [inline]`

Calls `base.setf(ios_base::oct, ios_base::basefield)`.

Definition at line 951 of file `ios_base.h`.

References `std::ios_base::basefield`, `std::ios_base::oct`, and `std::ios_base::setf()`.

4.11.4.126 `template<typename _Tp, typename _Alloc> bool std::operator!= (const deque<_Tp, _Alloc> & __x, const deque<_Tp, _Alloc> & __y) [inline]`

Based on `operator==`.

Definition at line 1943 of file `stl_deque.h`.

4.11.4.127 `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator!= (const basic_string<_CharT, _Traits, _Alloc> & __lhs, const basic_string<_CharT, _Traits, _Alloc> & __rhs) [inline]`

Test difference of two strings.

Parameters

lhs First string.

rhs Second string.

Returns

True if `lhs.compare(rhs) != 0`. False otherwise.

Definition at line 2473 of file `basic_string.h`.

4.11.4.128 `template<typename _CharT, typename _Traits, typename _Alloc
> bool std::operator!= (const _CharT * __lhs, const basic_string<
_CharT, _Traits, _Alloc > & __rhs) [inline]`

Test difference of C string and string.

Parameters

lhs C string.

rhs String.

Returns

True if *rhs.compare(lhs) != 0*. False otherwise.

Definition at line 2485 of file `basic_string.h`.

4.11.4.129 `template<typename _CharT, typename _Traits, typename _Alloc
> bool std::operator!= (const basic_string< _CharT, _Traits,
_Alloc > & __lhs, const _CharT * __rhs) [inline]`

Test difference of string and C string.

Parameters

lhs String.

rhs C string.

Returns

True if *lhs.compare(rhs) != 0*. False otherwise.

Definition at line 2497 of file `basic_string.h`.

4.11.4.130 `template<typename _Tp, typename _Alloc > bool std::operator!= (
const list< _Tp, _Alloc > & __x, const list< _Tp, _Alloc > & __y)
[inline]`

Based on `operator==`.

Definition at line 1600 of file `stl_list.h`.

4.11.4.131 `template<typename _Key , typename _Tp , typename _Compare ,
typename _Alloc > bool std::operator!=(const map< _Key, _Tp,
_Compare, _Alloc > & __x, const map< _Key, _Tp, _Compare,
_Alloc > & __y) [inline]`

Based on `operator==`.

Definition at line 901 of file `stl_map.h`.

4.11.4.132 `template<typename _Key , typename _Tp , typename _Compare ,
typename _Alloc > bool std::operator!=(const multimap< _Key,
_Tp, _Compare, _Alloc > & __x, const multimap< _Key, _Tp,
_Compare, _Alloc > & __y) [inline]`

Based on `operator==`.

Definition at line 819 of file `stl_multimap.h`.

4.11.4.133 `template<typename _Key , typename _Compare , typename _Alloc
> bool std::operator!=(const multiset< _Key, _Compare, _Alloc
> & __x, const multiset< _Key, _Compare, _Alloc > & __y)
[inline]`

Returns `!(x == y)`.

Definition at line 703 of file `stl_multiset.h`.

4.11.4.134 `template<class _T1 , class _T2 > constexpr bool std::operator!=(
const pair< _T1, _T2 > & __x, const pair< _T1, _T2 > & __y)
[inline]`

Uses `operator==` to find the result.

Definition at line 214 of file `stl_pair.h`.

4.11.4.135 `template<typename _Tp, typename _Seq > bool std::operator!=(
const queue< _Tp, _Seq > & __x, const queue< _Tp, _Seq > &
__y) [inline]`

Based on operator==.

Definition at line 290 of file stl_queue.h.

4.11.4.136 `template<typename _Key, typename _Compare, typename _Alloc
> bool std::operator!=(const set< _Key, _Compare, _Alloc > &
__x, const set< _Key, _Compare, _Alloc > & __y) [inline]`

Returns !(x == y).

Definition at line 720 of file stl_set.h.

4.11.4.137 `template<typename _Tp > bool std::operator!=(const
_Fwd_list_iterator< _Tp > & __x, const _Fwd_list_const_iterator<
_Tp > & __y) [inline]`

Forward list iterator inequality comparison.

Definition at line 265 of file forward_list.h.

4.11.4.138 `template<typename _Tp, typename _Alloc > bool std::operator!=(
const forward_list< _Tp, _Alloc > & __lx, const forward_list<
_Tp, _Alloc > & __ly) [inline]`

Based on operator==.

Definition at line 1268 of file forward_list.h.

4.11.4.139 `template<typename _Tp, typename _Alloc > bool std::operator!=(
const vector< _Tp, _Alloc > & __x, const vector< _Tp, _Alloc > &
__y) [inline]`

Based on operator==.

Definition at line 1297 of file `std_vector.h`.

4.11.4.140 `template<typename _Tp , typename _Seq > bool std::operator!= (`
`const stack< _Tp, _Seq > & __x, const stack< _Tp, _Seq > & __y`
`) [inline]`

Based on `operator==`.

Definition at line 265 of file `std_stack.h`.

4.11.4.141 `template<typename _Res , typename... _Args> bool std::operator!=`
`(const function< _Res(_Args...)> & __f, nullptr_t) [inline]`

Compares a polymorphic function object wrapper against 0 (the NULL pointer).

Returns

`false` if the wrapper has no target, `true` otherwise

This function will not throw an exception.

Definition at line 2232 of file `functional`.

4.11.4.142 `template<typename _Res , typename... _Args> bool std::operator!=`
`(nullptr_t, const function< _Res(_Args...)> & __f) [inline]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 2238 of file `functional`.

4.11.4.143 `template<size_t _Nb> bitset<_Nb> std::operator& (const bitset<`
`_Nb > & __x, const bitset< _Nb > & __y) [inline]`

Global bitwise operations on bitsets.

Parameters

x A bitset.

y A bitset of the same size as *x*.

Returns

A new bitset.

These should be self-explanatory.

Definition at line 1389 of file bitset.

4.11.4.144 `template<typename _CharT, typename _Traits, typename _Alloc
> basic_string< _CharT, _Traits, _Alloc > std::operator+ (_CharT
__lhs, const basic_string< _CharT, _Traits, _Alloc > & __rhs)`

Concatenate character and string.

Parameters

lhs First string.

rhs Last string.

Returns

New string with *lhs* followed by *rhs*.

Definition at line 710 of file basic_string.tcc.

References `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

4.11.4.145 `template<typename _CharT, typename _Traits, typename _Alloc
> basic_string< _CharT, _Traits, _Alloc> std::operator+ (const
basic_string< _CharT, _Traits, _Alloc > & __lhs, const _CharT *
__rhs) [inline]`

Concatenate string and C string.

Parameters

lhs First string.

rhs Last string.

Returns

New string with *lhs* followed by *rhs*.

Definition at line 2343 of file basic_string.h.

References `std::basic_string< _CharT, _Traits, _Alloc >::append()`.

4.11.4.146 `template<typename _CharT, typename _Traits, typename _Alloc
> basic_string<_CharT, _Traits, _Alloc> std::operator+ (const
basic_string< _CharT, _Traits, _Alloc > & __lhs, _CharT __rhs)
[inline]`

Concatenate string and character.

Parameters

lhs First string.

rhs Last string.

Returns

New string with *lhs* followed by *rhs*.

Definition at line 2359 of file basic_string.h.

4.11.4.147 `template<typename _CharT, typename _Traits, typename _Alloc
> basic_string<_CharT, _Traits, _Alloc> std::operator+ (const
basic_string< _CharT, _Traits, _Alloc > & __lhs, const
basic_string< _CharT, _Traits, _Alloc > & __rhs)`

Concatenate two strings.

Parameters

lhs First string.

rhs Last string.

Returns

New string with value of *lhs* followed by *rhs*.

Definition at line 2306 of file basic_string.h.

References `std::basic_string< _CharT, _Traits, _Alloc >::append()`.

4.11.4.148 `template<typename _CharT, typename _Traits, typename _Alloc
> basic_string< _CharT, _Traits, _Alloc > std::operator+ (const
_CharT * __lhs, const basic_string< _CharT, _Traits, _Alloc > &
__rhs)`

Concatenate C string and string.

Parameters

lhs First string.

rhs Last string.

Returns

New string with value of *lhs* followed by *rhs*.

Definition at line 694 of file basic_string.tcc.

References std::basic_string<_CharT, _Traits, _Alloc>::size().

4.11.4.149 `template<typename _Tp, typename _Alloc> bool std::operator< (const deque<_Tp, _Alloc> & __x, const deque<_Tp, _Alloc> & __y) [inline]`

Deque ordering relation.

Parameters

x A deque.

y A deque of the same type as *x*.

Returns

True iff *x* is lexicographically less than *y*.

This is a total ordering relation. It is linear in the size of the deques. The elements must be comparable with <.

See std::lexicographical_compare() for how the determination is made.

Definition at line 1935 of file stl_deque.h.

References lexicographical_compare().

4.11.4.150 `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator< (const basic_string<_CharT, _Traits, _Alloc> & __lhs, const basic_string<_CharT, _Traits, _Alloc> & __rhs) [inline]`

Test if string precedes string.

Parameters

lhs First string.

rhs Second string.

Returns

True if *lhs* precedes *rhs*. False otherwise.

Definition at line 2510 of file basic_string.h.

```
4.11.4.151 template<typename _CharT, typename _Traits, typename _Alloc  
> bool std::operator< ( const basic_string< _CharT, _Traits, _Alloc  
> & __lhs, const _CharT * __rhs ) [inline]
```

Test if string precedes C string.

Parameters

lhs String.

rhs C string.

Returns

True if *lhs* precedes *rhs*. False otherwise.

Definition at line 2522 of file basic_string.h.

```
4.11.4.152 template<typename _Tp, typename _Alloc > bool std::operator< (   
const list< _Tp, _Alloc > & __x, const list< _Tp, _Alloc > & __y )   
[inline]
```

List ordering relation.

Parameters

x A list.

y A list of the same type as *x*.

Returns

True iff *x* is lexicographically less than *y*.

This is a total ordering relation. It is linear in the size of the lists. The elements must be comparable with <.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 1593 of file `stl_list.h`.

References `lexicographical_compare()`.

```
4.11.4.153 template<typename _Key , typename _Tp , typename _Compare ,  
                typename _Alloc > bool std::operator< ( const map< _Key, _Tp,  
                _Compare, _Alloc > & __x, const map< _Key, _Tp, _Compare,  
                _Alloc > & __y ) [inline]
```

Map ordering relation.

Parameters

x A map.

y A map of the same type as *x*.

Returns

True iff *x* is lexicographically less than *y*.

This is a total ordering relation. It is linear in the size of the maps. The elements must be comparable with <.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 894 of file `stl_map.h`.

```
4.11.4.154 template<typename _Key , typename _Compare , typename _Alloc  
                > bool std::operator< ( const multiset< _Key, _Compare, _Alloc  
                > & __x, const multiset< _Key, _Compare, _Alloc > & __y )  
                [inline]
```

Multiset ordering relation.

Parameters

x A multiset.

y A multiset of the same type as *x*.

Returns

True iff x is lexicographically less than y .

This is a total ordering relation. It is linear in the size of the maps. The elements must be comparable with $<$.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 696 of file `stl_multiset.h`.

4.11.4.155 `template<class _T1, class _T2> constexpr bool std::operator< (`
`const pair< _T1, _T2 > & __x, const pair< _T1, _T2 > & __y)`
`[inline]`

[<http://gcc.gnu.org/onlinedocs/libstdc++/manual/utilities.html>](http://gcc.gnu.org/onlinedocs/libstdc++/manual/utilities.html)

Definition at line 207 of file `stl_pair.h`.

4.11.4.156 `template<typename _Tp, typename _Seq> bool std::operator< (`
`const queue< _Tp, _Seq > & __x, const queue< _Tp, _Seq > &`
`__y) [inline]`

Queue ordering relation.

Parameters

x A queue.

y A queue of the same type as x .

Returns

True iff x is lexicographically less than y .

This is an total ordering relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, the elements must be comparable with $<$, and `std::lexicographical_compare()` is usually used to make the determination.

Definition at line 284 of file `stl_queue.h`.

4.11.4.157 `template<typename _Key, typename _Compare, typename _Alloc
> bool std::operator< (const set< _Key, _Compare, _Alloc > &
__x, const set< _Key, _Compare, _Alloc > & __y) [inline]`

Set ordering relation.

Parameters

x A set.

y A set of the same type as *x*.

Returns

True iff *x* is lexicographically less than *y*.

This is a total ordering relation. It is linear in the size of the maps. The elements must be comparable with <.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 713 of file `stl_set.h`.

4.11.4.158 `template<typename _Tp, typename _Alloc > bool std::operator< (
const forward_list< _Tp, _Alloc > & __lx, const forward_list<
_Tp, _Alloc > & __ly) [inline]`

Forward list ordering relation.

Parameters

lx A `forward_list`.

ly A `forward_list` of the same type as *lx*.

Returns

True iff *lx* is lexicographically less than *ly*.

This is a total ordering relation. It is linear in the size of the forward lists. The elements must be comparable with <.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 1260 of file `forward_list.h`.

References `lexicographical_compare()`.

4.11.4.159 `template<typename _Tp, typename _Alloc> bool std::operator< (const vector< _Tp, _Alloc> & __x, const vector< _Tp, _Alloc> & __y) [inline]`

Vector ordering relation.

Parameters

- x* A vector.
- y* A vector of the same type as *x*.

Returns

True iff *x* is lexicographically less than *y*.

This is a total ordering relation. It is linear in the size of the vectors. The elements must be comparable with <.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 1290 of file `std_vector.h`.

References `lexicographical_compare()`.

4.11.4.160 `template<typename _Tp, typename _Seq> bool std::operator< (const stack< _Tp, _Seq> & __x, const stack< _Tp, _Seq> & __y) [inline]`

Stack ordering relation.

Parameters

- x* A stack.
- y* A stack of the same type as *x*.

Returns

True iff *x* is lexicographically less than *y*.

This is an total ordering relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, the elements must be comparable with <, and `std::lexicographical_compare()` is usually used to make the determination.

Definition at line 259 of file `std_stack.h`.

4.11.4.161 `template<typename _CharT, typename _Traits, typename _Alloc
> bool std::operator< (const _CharT * __lhs, const basic_string<
_CharT, _Traits, _Alloc > & __rhs) [inline]`

Test if C string precedes string.

Parameters

lhs C string.

rhs String.

Returns

True if *lhs* precedes *rhs*. False otherwise.

Definition at line 2534 of file basic_string.h.

References std::basic_string< _CharT, _Traits, _Alloc >::compare().

4.11.4.162 `template<typename _Key, typename _Tp, typename _Compare,
typename _Alloc > bool std::operator< (const multimap< _Key,
_Tp, _Compare, _Alloc > & __x, const multimap< _Key, _Tp,
_Compare, _Alloc > & __y) [inline]`

Multimap ordering relation.

Parameters

x A multimap.

y A multimap of the same type as *x*.

Returns

True iff *x* is lexicographically less than *y*.

This is a total ordering relation. It is linear in the size of the multimaps. The elements must be comparable with <.

See std::lexicographical_compare() for how the determination is made.

Definition at line 812 of file stl_multimap.h.

4.11.4.163 `template<class _CharT , class _Traits , size_t _Nb>
 std::basic_ostream<_CharT, _Traits>& std::operator<< (
 std::basic_ostream<_CharT, _Traits> & __os, const bitset<_Nb
 > & __x)`

Global I/O operators for bitsets.

Direct I/O between streams and bitsets is supported. Output is straightforward. Input will skip whitespace, only accept *0* and *1* characters, and will only extract as many digits as the bitset will hold.

Definition at line 1494 of file `bitset`.

References `std::__ctype_abstract_base<_CharT>::widen()`.

4.11.4.164 `template<typename _CharT , typename _Traits >
 basic_ostream<_CharT, _Traits>& std::operator<< (
 basic_ostream<_CharT, _Traits> & __out, _CharT __c)
 [inline]`

Character inserters.

Parameters

out An output stream.

c A character.

Returns

out

Behaves like one of the formatted arithmetic inserters described in [std::basic_ostream](#). After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2]). `out.width(0)` is then called.

If *c* is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 451 of file `ostream`.

```
4.11.4.165 template<typename _CharT, typename _Traits >
    basic_ostream<_CharT, _Traits>& std::operator<< (
        basic_ostream<_CharT, _Traits> & __out, char __c )
    [inline]
```

Character inserters.

Parameters

out An output stream.

c A character.

Returns

out

Behaves like one of the formatted arithmetic inserters described in [std::basic_ostream](#). After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `out.width(0)` is then called.

If *c* is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 456 of file `ostream`.

```
4.11.4.166 template<class _Traits > basic_ostream<char, _Traits>&
    std::operator<< ( basic_ostream< char, _Traits> & __out, char
        __c ) [inline]
```

Character inserters.

Parameters

out An output stream.

c A character.

Returns

out

Behaves like one of the formatted arithmetic inserters described in [std::basic_ostream](#). After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `out.width(0)` is then called.

If *c* is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 462 of file `ostream`.

```
4.11.4.167 template<class _Traits > basic_ostream<char, _Traits>&  
std::operator<<( basic_ostream< char, _Traits > & __out, signed  
char __c ) [inline]
```

Character inserters.

Parameters

out An output stream.

c A character.

Returns

out

Behaves like one of the formatted arithmetic inserters described in [std::basic_ostream](#). After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `out.width(0)` is then called.

If *c* is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 468 of file `ostream`.

```
4.11.4.168 template<class _Traits > basic_ostream<char, _Traits>&  
std::operator<<( basic_ostream< char, _Traits > & __out, const  
char * __s ) [inline]
```

String inserters.

Parameters

out An output stream.

s A character string.

Returns

out

Precondition

s must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in [std::basic_ostream](#). After constructing a sentry object with good status, this function inserts `traits::length(s)` characters starting at *s*, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `out.width(0)` is then called.

Definition at line 510 of file ostream.

References `std::ios_base::badbit`.

4.11.4.169 `template<class _Traits > basic_ostream<char, _Traits>&
std::operator<<(basic_ostream< char, _Traits > & __out, const
signed char * __s) [inline]`

String inserters.

Parameters

out An output stream.

s A character string.

Returns

out

Precondition

s must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in [std::basic_ostream](#). After constructing a sentry object with good status, this function inserts `traits::length(s)` characters starting at *s*, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `out.width(0)` is then called.

Definition at line 523 of file ostream.

4.11.4.170 `template<class _Traits > basic_ostream<char, _Traits>&
std::operator<<(basic_ostream< char, _Traits > & __out, const
unsigned char * __s) [inline]`

String inserters.

Parameters

out An output stream.
s A character string.

Returns

out

Precondition

s must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in [std::basic_ostream](#). After constructing a sentry object with good status, this function inserts `traits::length(s)` characters starting at *s*, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `out.width(0)` is then called.

Definition at line 528 of file ostream.

```
4.11.4.171 template<typename _CharT, typename _Traits >
    basic_ostream<_CharT, _Traits>& std::operator<< (
    basic_ostream<_CharT, _Traits> & __out, const _CharT * __s )
    [inline]
```

String inserters.

Parameters

out An output stream.
s A character string.

Returns

out

Precondition

s must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in [std::basic_ostream](#). After constructing a sentry object with good status, this function inserts `traits::length(s)` characters starting at *s*, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `out.width(0)` is then called.

Definition at line 493 of file ostream.

References `std::ios_base::badbit`.

4.11.4.172 `template<typename _CharT, typename _Traits > basic_ostream<_CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > & __out, const char * __s)`

String inserters.

Parameters

out An output stream.

s A character string.

Returns

out

Precondition

s must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in [std::basic_ostream](#). After constructing a sentry object with good status, this function inserts `traits::length(s)` characters starting at *s*, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `out.width(0)` is then called.

Definition at line 323 of file ostream.tcc.

References `std::ios_base::badbit`.

4.11.4.173 `template<typename _CharT, typename _Traits, typename _Tp > basic_ostream<_CharT, _Traits>& std::operator<< (basic_ostream< _CharT, _Traits > && __os, const _Tp & __x) [inline]`

Generic inserter for rvalue stream.

Parameters

os An input stream.

x A reference to the object being inserted.

Returns

os

This is just a forwarding function to allow insertion to rvalue streams since they won't bind to the inserter functions that take an lvalue reference.

Definition at line 581 of file ostream.

```
4.11.4.174 template<class _Traits > basic_ostream<char, _Traits>&  
std::operator<< ( basic_ostream< char, _Traits > & __out,  
unsigned char __c ) [inline]
```

Character inserters.

Parameters

out An output stream.

c A character.

Returns

out

Behaves like one of the formatted arithmetic inserters described in [std::basic_ostream](#). After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `out.width(0)` is then called.

If *c* is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 473 of file ostream.

```
4.11.4.175 template<typename _CharT , typename _Traits , typename  
_Alloc > basic_ostream<_CharT, _Traits>& std::operator<< (  
basic_ostream< _CharT, _Traits > & __os, const basic_string<  
_CharT, _Traits, _Alloc > & __str ) [inline]
```

Write string to a stream.

Parameters

os Output stream.

str String to write out.

Returns

Reference to the output stream.

Output characters of *str* into *os* following the same rules as for writing a C string.

Definition at line 2693 of file `basic_string.h`.

```
4.11.4.176 template<typename _CharT, typename _Traits, typename
               _Alloc, template< typename, typename, typename > class
               _Base> basic_ostream<_CharT, _Traits>& std::operator<<
               ( basic_ostream<_CharT, _Traits> & __os, const
               __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base> &
               __str ) [inline]
```

Write string to a stream.

Parameters

__os Output stream.

__str String to write out.

Returns

Reference to the output stream.

Output characters of *__str* into *os* following the same rules as for writing a C string.

Definition at line 2463 of file `vstring.h`.

```
4.11.4.177 template<typename _Tp, typename _Alloc> bool std::operator<=
               ( const deque<_Tp, _Alloc> & __x, const deque<_Tp, _Alloc>
               & __y ) [inline]
```

Based on `operator<`.

Definition at line 1957 of file `stl_deque.h`.

```
4.11.4.178 template<typename _CharT, typename _Traits, typename _Alloc
               > bool std::operator<= ( const basic_string<_CharT, _Traits,
               _Alloc> & __lhs, const _CharT* __rhs ) [inline]
```

Test if string doesn't follow C string.

Parameters

lhs String.

rhs C string.

Returns

True if *lhs* doesn't follow *rhs*. False otherwise.

Definition at line 2596 of file basic_string.h.

4.11.4.179 `template<typename _CharT, typename _Traits, typename
_Alloc> bool std::operator<= (const _CharT * __lhs, const
basic_string< _CharT, _Traits, _Alloc> & __rhs) [inline]`

Test if C string doesn't follow string.

Parameters

lhs C string.

rhs String.

Returns

True if *lhs* doesn't follow *rhs*. False otherwise.

Definition at line 2608 of file basic_string.h.

References std::basic_string< _CharT, _Traits, _Alloc>::compare().

4.11.4.180 `template<typename _Tp, typename _Alloc> bool std::operator<= (const list< _Tp, _Alloc> & __x, const list< _Tp, _Alloc> & __y) [inline]`

Based on operator<.

Definition at line 1612 of file stl_list.h.

4.11.4.181 `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc> bool std::operator<= (const map< _Key, _Tp, _Compare, _Alloc> & __x, const map< _Key, _Tp, _Compare, _Alloc> & __y) [inline]`

Based on operator<.

Definition at line 915 of file stl_map.h.

4.11.4.182 `template<typename _Key , typename _Tp , typename _Compare ,
typename _Alloc > bool std::operator<= (const multimap< _Key,
_Tp, _Compare, _Alloc > & __x, const multimap< _Key, _Tp,
_Compare, _Alloc > & __y) [inline]`

Based on operator<.

Definition at line 833 of file stl_multimap.h.

4.11.4.183 `template<typename _Key , typename _Compare , typename _Alloc
> bool std::operator<= (const multiset< _Key, _Compare, _Alloc
> & __x, const multiset< _Key, _Compare, _Alloc > & __y)
[inline]`

Returns !(y < x).

Definition at line 717 of file stl_multiset.h.

4.11.4.184 `template<class _T1 , class _T2 > constexpr bool std::operator<= (
const pair< _T1, _T2 > & __x, const pair< _T1, _T2 > & __y)
[inline]`

Uses operator< to find the result.

Definition at line 226 of file stl_pair.h.

4.11.4.185 `template<typename _Tp , typename _Seq > bool std::operator<= (
const queue< _Tp, _Seq > & __x, const queue< _Tp, _Seq > &
__y) [inline]`

Based on operator<.

Definition at line 302 of file stl_queue.h.

4.11.4.186 `template<typename _Key , typename _Compare , typename _Alloc
> bool std::operator<= (const set< _Key, _Compare, _Alloc > &
__x, const set< _Key, _Compare, _Alloc > & __y) [inline]`

Returns $!(y < x)$.

Definition at line 734 of file `stl_set.h`.

```
4.11.4.187  template<typename _CharT, typename _Traits, typename _Alloc
              > bool std::operator<= ( const basic_string< _CharT, _Traits,
              _Alloc > & __lhs, const basic_string< _CharT, _Traits, _Alloc > &
              __rhs ) [inline]
```

Test if string doesn't follow string.

Parameters

lhs First string.

rhs Second string.

Returns

True if *lhs* doesn't follow *rhs*. False otherwise.

Definition at line 2584 of file `basic_string.h`.

```
4.11.4.188  template<typename _Tp, typename _Seq > bool std::operator<= (
              const stack< _Tp, _Seq > & __x, const stack< _Tp, _Seq > & __y
              ) [inline]
```

Based on `operator<`.

Definition at line 277 of file `stl_stack.h`.

```
4.11.4.189  template<typename _Tp, typename _Alloc > bool std::operator<=
              ( const vector< _Tp, _Alloc > & __x, const vector< _Tp, _Alloc >
              & __y ) [inline]
```

Based on `operator<`.

Definition at line 1309 of file `stl_vector.h`.

4.11.4.190 `template<typename _Tp , typename _Alloc > bool std::operator<=`
`(const forward_list< _Tp, _Alloc > & __lx, const forward_list<`
`_Tp, _Alloc > & __ly) [inline]`

Based on operator<.

Definition at line 1289 of file forward_list.h.

4.11.4.191 `template<typename _Res , typename... _Args> bool`
`std::operator==(nullptr_t, const function< _Res(_Args...)> &`
`__f) [inline]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 2220 of file functional.

4.11.4.192 `template<typename _Res , typename... _Args> bool`
`std::operator==(const function< _Res(_Args...)> & __f, nullptr_t`
`) [inline]`

Compares a polymorphic function object wrapper against 0 (the NULL pointer).

Returns

`true` if the wrapper has no target, `false` otherwise

This function will not throw an exception.

Definition at line 2214 of file functional.

4.11.4.193 `template<typename _Key , typename _Compare , typename _Alloc`
`> bool std::operator==(const set< _Key, _Compare, _Alloc > &`
`__x, const set< _Key, _Compare, _Alloc > & __y) [inline]`

Set equality comparison.

Parameters

x A set.

y A set of the same type as *x*.

Returns

True iff the size and elements of the sets are equal.

This is an equivalence relation. It is linear in the size of the sets. Sets are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 696 of file stl_set.h.

```
4.11.4.194 template<typename _Key , typename _Tp , typename _Compare ,  
                typename _Alloc > bool std::operator==( const map< _Key, _Tp,  
                _Compare, _Alloc > & __x, const map< _Key, _Tp, _Compare,  
                _Alloc > & __y ) [inline]
```

Map equality comparison.

Parameters

x A map.

y A map of the same type as *x*.

Returns

True iff the size and elements of the maps are equal.

This is an equivalence relation. It is linear in the size of the maps. Maps are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 877 of file stl_map.h.

```
4.11.4.195 template<typename _Key , typename _Tp , typename _Compare ,  
                typename _Alloc > bool std::operator==( const multimap< _Key,  
                _Tp, _Compare, _Alloc > & __x, const multimap< _Key, _Tp,  
                _Compare, _Alloc > & __y ) [inline]
```

Multimap equality comparison.

Parameters

x A multimap.

y A multimap of the same type as *x*.

Returns

True iff the size and elements of the maps are equal.

This is an equivalence relation. It is linear in the size of the multimaps. Multimaps are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 795 of file `std_multimap.h`.

4.11.4.196 `template<typename _Tp, typename _Alloc> bool std::operator==(const deque< _Tp, _Alloc> & __x, const deque< _Tp, _Alloc> & __y) [inline]`

Deque equality comparison.

Parameters

- x* A deque.
- y* A deque of the same type as *x*.

Returns

True iff the size and elements of the deques are equal.

This is an equivalence relation. It is linear in the size of the deques. Deques are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 1917 of file `std_deque.h`.

References `std::deque< _Tp, _Alloc>::begin()`, `std::deque< _Tp, _Alloc>::end()`, `equal()`, and `std::deque< _Tp, _Alloc>::size()`.

4.11.4.197 `template<typename _Tp> bool std::operator==(const _Fwd_list_iterator< _Tp> & __x, const _Fwd_list_const_iterator< _Tp> & __y) [inline]`

Forward list iterator equality comparison.

Definition at line 256 of file `forward_list.h`.

4.11.4.198 `template<typename _Key, typename _Compare, typename _Alloc> bool std::operator==(const multiset< _Key, _Compare, _Alloc> & __x, const multiset< _Key, _Compare, _Alloc> & __y) [inline]`

Multiset equality comparison.

Parameters

- x* A multiset.
- y* A multiset of the same type as *x*.

Returns

True iff the size and elements of the multisets are equal.

This is an equivalence relation. It is linear in the size of the multisets. Multisets are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 679 of file stl_multiset.h.

```
4.11.4.199 template<typename _Tp , typename _Seq > bool std::operator== (
    const stack< _Tp, _Seq > & __x, const stack< _Tp, _Seq > & __y
) [inline]
```

Stack equality comparison.

Parameters

- x* A stack.
- y* A stack of the same type as *x*.

Returns

True iff the size and elements of the stacks are equal.

This is an equivalence relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, and stacks are considered equivalent if their sequences compare equal.

Definition at line 241 of file stl_stack.h.

```
4.11.4.200 template<class _T1 , class _T2 > constexpr bool std::operator== (
    const pair< _T1, _T2 > & __x, const pair< _T1, _T2 > & __y )
[inline]
```

Two pairs of the same type are equal iff their members are equal.

Definition at line 201 of file `std_pair.h`.

References `std::pair<_T1, _T2>::first`, and `std::pair<_T1, _T2>::second`.

4.11.4.201 `template<typename _Tp, typename _Alloc> bool std::operator==(const forward_list<_Tp, _Alloc> & __lx, const forward_list<_Tp, _Alloc> & __ly)`

Forward list equality comparison.

Parameters

lx A `forward_list`

ly A `forward_list` of the same type as *lx*.

Returns

True iff the size and elements of the forward lists are equal.

This is an equivalence relation. It is linear in the size of the forward lists. Deques are considered equivalent if corresponding elements compare equal.

Definition at line 379 of file `forward_list.tcc`.

References `std::forward_list<_Tp, _Alloc>::cbegin()`, and `std::forward_list<_Tp, _Alloc>::cend()`.

4.11.4.202 `template<typename _Tp, typename _Seq> bool std::operator==(const queue<_Tp, _Seq> & __x, const queue<_Tp, _Seq> & __y) [inline]`

Queue equality comparison.

Parameters

x A queue.

y A queue of the same type as *x*.

Returns

True iff the size and elements of the queues are equal.

This is an equivalence relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, and queues are considered equivalent if their sequences compare equal.

Definition at line 266 of file `std_queue.h`.

References `std::queue<_Tp, _Sequence>::c`.

4.11.4.203 `template<typename _StateT> bool std::operator==(const fpos<_StateT> & __lhs, const fpos<_StateT> & __rhs) [inline]`

Test if equivalent to another position.

Definition at line 218 of file `postypes.h`.

4.11.4.204 `template<typename _Tp, typename _Alloc> bool std::operator==(const vector<_Tp, _Alloc> & __x, const vector<_Tp, _Alloc> & __y) [inline]`

Vector equality comparison.

Parameters

x A vector.

y A vector of the same type as *x*.

Returns

True iff the size and elements of the vectors are equal.

This is an equivalence relation. It is linear in the size of the vectors. Vectors are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 1273 of file `std_vector.h`.

References `std::vector<_Tp, _Alloc>::begin()`, `std::vector<_Tp, _Alloc>::end()`, `equal()`, and `std::vector<_Tp, _Alloc>::size()`.

4.11.4.205 `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator==(const basic_string<_CharT, _Traits, _Alloc> & __lhs, const _CharT* __rhs) [inline]`

Test equivalence of string and C string.

Parameters

lhs String.
rhs C string.

Returns

True if *lhs.compare(rhs) == 0*. False otherwise.

Definition at line 2460 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

4.11.4.206 `template<typename _CharT, typename _Traits, typename _Alloc
> bool std::operator==(const basic_string< _CharT, _Traits,
_Alloc > & __lhs, const basic_string< _CharT, _Traits, _Alloc > &
__rhs) [inline]`

Test equivalence of two strings.

Parameters

lhs First string.
rhs Second string.

Returns

True if *lhs.compare(rhs) == 0*. False otherwise.

Definition at line 2427 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

4.11.4.207 `template<typename _Tp, typename _Alloc > bool std::operator==(
const list< _Tp, _Alloc > & __x, const list< _Tp, _Alloc > & __y)
[inline]`

List equality comparison.

Parameters

x A list.
y A list of the same type as *x*.

Returns

True iff the size and elements of the lists are equal.

This is an equivalence relation. It is linear in the size of the lists. Lists are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 1564 of file `stl_list.h`.

References `std::list<_Tp, _Alloc>::begin()`, and `std::list<_Tp, _Alloc>::end()`.

4.11.4.208 `template<typename _CharT, typename _Traits, typename
_Alloc> bool std::operator==(const _CharT * __lhs, const
basic_string<_CharT, _Traits, _Alloc> & __rhs) [inline]`

Test equivalence of C string and string.

Parameters

lhs C string.

rhs String.

Returns

True if `rhs.compare(lhs) == 0`. False otherwise.

Definition at line 2448 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

4.11.4.209 `template<typename _Key, typename _Tp, typename _Compare,
typename _Alloc> bool std::operator> (const map<_Key, _Tp,
_Compare, _Alloc> & __x, const map<_Key, _Tp, _Compare,
_Alloc> & __y) [inline]`

Based on `operator<`.

Definition at line 908 of file `stl_map.h`.

4.11.4.210 `template<typename _Tp, typename _Seq> bool std::operator> (`
`const queue<_Tp, _Seq> & __x, const queue<_Tp, _Seq> &`
`__y) [inline]`

Based on `operator<`.

Definition at line 296 of file `stl_queue.h`.

4.11.4.211 `template<typename _CharT, typename _Traits, typename _Alloc
> bool std::operator> (const _CharT * __lhs, const basic_string<
_CharT, _Traits, _Alloc > & __rhs) [inline]`

Test if C string follows string.

Parameters

lhs C string.

rhs String.

Returns

True if *lhs* follows *rhs*. False otherwise.

Definition at line 2571 of file `basic_string.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::compare()`.

4.11.4.212 `template<typename _Tp, typename _Alloc > bool std::operator> (
const deque< _Tp, _Alloc > & __x, const deque< _Tp, _Alloc > &
__y) [inline]`

Based on `operator<`.

Definition at line 1950 of file `stl_deque.h`.

4.11.4.213 `template<class _T1, class _T2 > constexpr bool std::operator> (
const pair< _T1, _T2 > & __x, const pair< _T1, _T2 > & __y)
[inline]`

Uses `operator<` to find the result.

Definition at line 220 of file `stl_pair.h`.

4.11.4.214 `template<typename _CharT, typename _Traits, typename _Alloc
> bool std::operator> (const basic_string< _CharT, _Traits, _Alloc
> & __lhs, const basic_string< _CharT, _Traits, _Alloc > & __rhs
) [inline]`

Test if string follows string.

Parameters

lhs First string.

rhs Second string.

Returns

True if *lhs* follows *rhs*. False otherwise.

Definition at line 2547 of file `basic_string.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::compare()`.

4.11.4.215 `template<typename _Key, typename _Tp, typename _Compare ,
typename _Alloc > bool std::operator> (const multimap< _Key,
_Tp, _Compare, _Alloc > & __x, const multimap< _Key, _Tp,
_Compare, _Alloc > & __y) [inline]`

Based on `operator<`.

Definition at line 826 of file `stl_multimap.h`.

4.11.4.216 `template<typename _Key, typename _Compare, typename _Alloc
> bool std::operator> (const set< _Key, _Compare, _Alloc > &
__x, const set< _Key, _Compare, _Alloc > & __y) [inline]`

Returns $y < x$.

Definition at line 727 of file `stl_set.h`.

4.11.4.217 `template<typename _Key , typename _Compare , typename _Alloc
> bool std::operator> (const multiset< _Key, _Compare, _Alloc
> & __x, const multiset< _Key, _Compare, _Alloc > & __y)
[inline]`

Returns $y < x$.

Definition at line 710 of file `stl_multiset.h`.

4.11.4.218 `template<typename _Tp , typename _Seq > bool std::operator> (
const stack< _Tp, _Seq > & __x, const stack< _Tp, _Seq > & __y
) [inline]`

Based on `operator<`.

Definition at line 271 of file `stl_stack.h`.

4.11.4.219 `template<typename _Tp , typename _Alloc > bool std::operator> (
const vector< _Tp, _Alloc > & __x, const vector< _Tp, _Alloc > &
__y) [inline]`

Based on `operator<`.

Definition at line 1303 of file `stl_vector.h`.

4.11.4.220 `template<typename _CharT , typename _Traits , typename _Alloc
> bool std::operator> (const basic_string< _CharT, _Traits, _Alloc
> & __lhs, const _CharT * __rhs) [inline]`

Test if string follows C string.

Parameters

lhs String.

rhs C string.

Returns

True if *lhs* follows *rhs*. False otherwise.

Definition at line 2559 of file basic_string.h.

References std::basic_string<_CharT, _Traits, _Alloc>::compare().

4.11.4.221 `template<typename _Tp, typename _Alloc> bool std::operator> (
 const forward_list< _Tp, _Alloc> & __x, const forward_list<
 _Tp, _Alloc> & __y) [inline]`

Based on operator<.

Definition at line 1275 of file forward_list.h.

4.11.4.222 `template<typename _Tp, typename _Alloc> bool std::operator> (
 const list< _Tp, _Alloc> & __x, const list< _Tp, _Alloc> & __y)
 [inline]`

Based on operator<.

Definition at line 1606 of file stl_list.h.

4.11.4.223 `template<typename _Key, typename _Tp, typename _Compare,
 typename _Alloc> bool std::operator>= (const map< _Key, _Tp,
 _Compare, _Alloc> & __x, const map< _Key, _Tp, _Compare,
 _Alloc> & __y) [inline]`

Based on operator<.

Definition at line 922 of file stl_map.h.

4.11.4.224 `template<typename _Key, typename _Compare, typename _Alloc
> bool std::operator>= (const multiset< _Key, _Compare, _Alloc
> & __x, const multiset< _Key, _Compare, _Alloc> & __y)
 [inline]`

Returns !(x < y).

Definition at line 724 of file stl_multiset.h.

4.11.4.225 `template<typename _Tp, typename _Alloc > bool std::operator>=`
`(const deque< _Tp, _Alloc > & __x, const deque< _Tp, _Alloc >`
`& __y) [inline]`

Based on operator<.

Definition at line 1964 of file stl_deque.h.

4.11.4.226 `template<typename _CharT, typename _Traits, typename`
`_Alloc > bool std::operator>= (const _CharT * __lhs, const`
`basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline]`

Test if C string doesn't precede string.

Parameters

lhs C string.

rhs String.

Returns

True if *lhs* doesn't precede *rhs*. False otherwise.

Definition at line 2645 of file basic_string.h.

References std::basic_string< _CharT, _Traits, _Alloc >::compare().

4.11.4.227 `template<typename _Tp, typename _Alloc > bool std::operator>=`
`(const forward_list< _Tp, _Alloc > & __lx, const forward_list<`
`_Tp, _Alloc > & __ly) [inline]`

Based on operator<.

Definition at line 1282 of file forward_list.h.

4.11.4.228 `template<typename _Tp, typename _Alloc > bool std::operator>=`
`(const list< _Tp, _Alloc > & __x, const list< _Tp, _Alloc > & __y`
`) [inline]`

Based on operator<.

Definition at line 1618 of file stl_list.h.

4.11.4.229 `template<typename _Key , typename _Tp , typename _Compare ,
typename _Alloc > bool std::operator>= (const multimap< _Key,
_Tp, _Compare, _Alloc > & __x, const multimap< _Key, _Tp,
_Compare, _Alloc > & __y) [inline]`

Based on operator<.

Definition at line 840 of file stl_multimap.h.

4.11.4.230 `template<typename _CharT , typename _Traits , typename _Alloc
> bool std::operator>= (const basic_string< _CharT, _Traits,
_Alloc > & __lhs, const basic_string< _CharT, _Traits, _Alloc > &
__rhs) [inline]`

Test if string doesn't precede string.

Parameters

lhs First string.

rhs Second string.

Returns

True if *lhs* doesn't precede *rhs*. False otherwise.

Definition at line 2621 of file basic_string.h.

References std::basic_string< _CharT, _Traits, _Alloc >::compare().

4.11.4.231 `template<typename _CharT , typename _Traits , typename _Alloc
> bool std::operator>= (const basic_string< _CharT, _Traits,
_Alloc > & __lhs, const _CharT * __rhs) [inline]`

Test if string doesn't precede C string.

Parameters

lhs String.

rhs C string.

Returns

True if *lhs* doesn't precede *rhs*. False otherwise.

Definition at line 2633 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

4.11.4.232 `template<typename _Key, typename _Compare, typename _Alloc
> bool std::operator>= (const set< _Key, _Compare, _Alloc > &
__x, const set< _Key, _Compare, _Alloc > & __y) [inline]`

Returns `!(x < y)`.

Definition at line 741 of file `stl_set.h`.

4.11.4.233 `template<typename _Tp, typename _Seq > bool std::operator>= (
const stack< _Tp, _Seq > & __x, const stack< _Tp, _Seq > & __y
) [inline]`

Based on `operator<`.

Definition at line 283 of file `stl_stack.h`.

4.11.4.234 `template<typename _Tp, typename _Alloc > bool std::operator>= (
const vector< _Tp, _Alloc > & __x, const vector< _Tp, _Alloc >
& __y) [inline]`

Based on `operator<`.

Definition at line 1315 of file `stl_vector.h`.

4.11.4.235 `template<class _T1, class _T2 > constexpr bool std::operator>= (
const pair< _T1, _T2 > & __x, const pair< _T1, _T2 > & __y)
[inline]`

Uses `operator<` to find the result.

Definition at line 232 of file `stl_pair.h`.

4.11.4.236 `template<typename _Tp, typename _Seq > bool std::operator>= (
 const queue< _Tp, _Seq > & __x, const queue< _Tp, _Seq > &
 __y) [inline]`

Based on operator<.

Definition at line 308 of file stl_queue.h.

4.11.4.237 `template<class _Traits > basic_istream<char, _Traits>&
 std::operator>> (basic_istream< char, _Traits > & __in,
 unsigned char & __c) [inline]`

Character extractors.

Parameters

in An input stream.

c A character reference.

Returns

in

Behaves like one of the formatted arithmetic extractors described in [std::basic_istream](#). After constructing a sentry object with good status, this function extracts a character (if one is available) and stores it in *c*. Otherwise, sets failbit in the input stream.

Definition at line 706 of file istream.

4.11.4.238 `template<class _CharT, class _Traits, size_t _Nb>
 std::basic_istream<_CharT, _Traits>& std::operator>> (
 std::basic_istream< _CharT, _Traits > & __is, bitset< _Nb > &
 __x)`

Global I/O operators for bitsets.

Direct I/O between streams and bitsets is supported. Output is straightforward. Input will skip whitespace, only accept *0* and *1* characters, and will only extract as many digits as the bitset will hold.

Definition at line 1426 of file bitset.

References `std::basic_string< _CharT, _Traits, _Alloc >::empty()`, `std::basic_string< _CharT, _Traits, _Alloc >::push_back()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`,

`std::basic_string< _CharT, _Traits, _Alloc >::reserve()`, `std::basic_ios< _CharT, _Traits >::setstate()`, and `std::basic_ios< _CharT, _Traits >::widen()`.

4.11.4.239 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > & __in, _CharT * __s)`

Character string extractors.

Parameters

- in* An input stream.
- s* A pointer to a character array.

Returns

in

Behaves like one of the formatted arithmetic extractors described in [std::basic_istream](#). After constructing a sentry object with good status, this function extracts up to *n* characters and stores them into the array starting at *s*. *n* is defined as:

- if `width()` is greater than zero, *n* is `width()` otherwise
- *n* is *the number of elements of the largest array of **
- *char_type that can store a terminating eos.*
- [27.6.1.2.3]/6

Characters are extracted and stored until one of the following happens:

- *n*-1 characters are stored
- EOF is reached
- the next character is whitespace according to the current locale
- the next character is a null byte (i.e., `charT()`)

`width(0)` is then called for the input stream.

If no characters are extracted, sets failbit.

Definition at line 957 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::getloc()`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, and `std::ios_base::width()`.

4.11.4.240 `template<> basic_istream<char>& std::operator>> (`
`basic_istream< char > & __in, char * __s)`

Character string extractors.

Parameters

- in* An input stream.
- s* A pointer to a character array.

Returns

in

Behaves like one of the formatted arithmetic extractors described in [std::basic_istream](#). After constructing a sentry object with good status, this function extracts up to *n* characters and stores them into the array starting at *s*. *n* is defined as:

- if `width()` is greater than zero, *n* is `width()` otherwise
- *n* is *the number of elements of the largest array of **
- *char_type that can store a terminating eos.*
- [27.6.1.2.3]/6

Characters are extracted and stored until one of the following happens:

- *n*-1 characters are stored
- EOF is reached
- the next character is whitespace according to the current locale
- the next character is a null byte (i.e., `charT()`)

`width(0)` is then called for the input stream.

If no characters are extracted, sets failbit.

4.11.4.241 `template<class _Traits > basic_istream<char, _Traits>&`
`std::operator>> (basic_istream< char, _Traits > & __in,`
`unsigned char * __s) [inline]`

Character string extractors.

Parameters

- in* An input stream.
- s* A pointer to a character array.

Returns

in

Behaves like one of the formatted arithmetic extractors described in [std::basic_istream](#). After constructing a sentry object with good status, this function extracts up to *n* characters and stores them into the array starting at *s*. *n* is defined as:

- if `width()` is greater than zero, *n* is `width()` otherwise
- *n* is *the number of elements of the largest array of **
- *char_type* that can store a terminating *eos*.
- `[27.6.1.2.3]/6`

Characters are extracted and stored until one of the following happens:

- *n*-1 characters are stored
- EOF is reached
- the next character is whitespace according to the current locale
- the next character is a null byte (i.e., `charT()`)

`width(0)` is then called for the input stream.

If no characters are extracted, sets failbit.

Definition at line 753 of file `istream`.

4.11.4.242 `template<class _Traits> basic_istream<char, _Traits>&
std::operator>>(basic_istream< char, _Traits> & __in, signed
char * __s) [inline]`

Character string extractors.

Parameters

- in* An input stream.
- s* A pointer to a character array.

Returns

in

Behaves like one of the formatted arithmetic extractors described in [std::basic_istream](#). After constructing a sentry object with good status, this function extracts up to *n* characters and stores them into the array starting at *s*. *n* is defined as:

- if `width()` is greater than zero, *n* is `width()` otherwise
- *n* is *the number of elements of the largest array of **
- *char_type that can store a terminating eos.*
- [27.6.1.2.3]/6

Characters are extracted and stored until one of the following happens:

- *n*-1 characters are stored
- EOF is reached
- the next character is whitespace according to the current locale
- the next character is a null byte (i.e., `charT()`)

`width(0)` is then called for the input stream.

If no characters are extracted, sets failbit.

Definition at line 758 of file `istream`.

4.11.4.243 `template<typename _CharT, typename _Traits, typename
_Tp> basic_istream<_CharT, _Traits>& std::operator>>(
basic_istream<_CharT, _Traits> && __is, _Tp & __x)
[inline]`

Generic extractor for rvalue stream.

Parameters

is An input stream.

x A reference to the extraction target.

Returns

is

This is just a forwarding function to allow extraction from rvalue streams since they won't bind to the extractor functions that take an lvalue reference.

Definition at line 849 of file istream.

```
4.11.4.244 template<typename _CharT, typename _Traits, typename _Alloc
> basic_istream<_CharT, _Traits> & std::operator>> (
    basic_istream<_CharT, _Traits> & __is, basic_string<_CharT,
    _Traits, _Alloc> & __str )
```

Read stream into a string.

Parameters

is Input stream.

str Buffer to store into.

Returns

Reference to the input stream.

Stores characters from *is* into *str* until whitespace is found, the end of the stream is encountered, or *str.max_size()* is reached. If *is.width()* is non-zero, that is the limit on the number of characters stored into *str*. Any previous contents of *str* are erased.

Definition at line 998 of file basic_string.tcc.

References `std::basic_string<_CharT, _Traits, _Alloc>::append()`, `std::basic_string<_CharT, _Traits, _Alloc>::erase()`, `std::ios_base::getloc()`, `std::basic_string<_CharT, _Traits, _Alloc>::max_size()`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::ios_base::width()`.

```
4.11.4.245 template<class _Traits> basic_istream<char, _Traits>&
    std::operator>> ( basic_istream<char, _Traits> & __in, signed
    char & __c ) [inline]
```

Character extractors.

Parameters

in An input stream.

c A character reference.

Returns

in

Behaves like one of the formatted arithmetic extractors described in [std::basic_istream](#). After constructing a sentry object with good status, this function extracts a character (if one is available) and stores it in *c*. Otherwise, sets failbit in the input stream.

Definition at line 711 of file istream.

4.11.4.246 `template<typename _CharT, typename _Traits, typename
_Alloc, template< typename, typename, typename
> class _Base> basic_istream< _CharT, _Traits > &
std::operator>> (basic_istream< _CharT, _Traits > & __is,
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &
__str)`

Read stream into a string.

Parameters

__is Input stream.

__str Buffer to store into.

Returns

Reference to the input stream.

Stores characters from *__is* into *__str* until whitespace is found, the end of the stream is encountered, or *str.max_size()* is reached. If *is.width()* is non-zero, that is the limit on the number of characters stored into *__str*. Any previous contents of *__str* are erased.

Definition at line 552 of file vstring.tcc.

References `std::ios_base::getloc()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, and `std::ios_base::width()`.

4.11.4.247 `template<typename _CharT, typename _Traits > basic_istream<
_CharT, _Traits > & std::operator>> (basic_istream< _CharT,
_Traits > & __in, _CharT & __c)`

Character extractors.

Parameters

in An input stream.

c A character reference.

Returns

in

Behaves like one of the formatted arithmetic extractors described in [std::basic_istream](#). After constructing a sentry object with good status, this function extracts a character (if one is available) and stores it in *c*. Otherwise, sets failbit in the input stream.

Definition at line 925 of file istream.tcc.

References `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

4.11.4.248 `template<size_t _Nb> bitset<_Nb> std::operator^ (const bitset<_Nb> & __x, const bitset<_Nb> & __y) [inline]`

Global bitwise operations on bitsets.

Parameters

x A bitset.

y A bitset of the same size as *x*.

Returns

A new bitset.

These should be self-explanatory.

Definition at line 1407 of file bitset.

4.11.4.249 `template<size_t _Nb> bitset<_Nb> std::operator| (const bitset<_Nb> & __x, const bitset<_Nb> & __y) [inline]`

Global bitwise operations on bitsets.

Parameters

x A bitset.

y A bitset of the same size as *x*.

Returns

A new bitset.

These should be self-explanatory.

Definition at line 1398 of file bitset.

4.11.4.250 `template<typename _InputIterator , typename _OutputIterator
> _OutputIterator std::partial_sum (_InputIterator __first,
_InputIterator __last, _OutputIterator __result)`

Return list of partial sums.

Accumulates the values in the range [first,last) using operator+(). As each successive input value is added into the total, that partial sum is written to *result*. Therefore, the first value in result is the first value of the input, the second value in result is the sum of the first and second input values, and so on.

Parameters

first Start of input range.

last End of input range.

result Output to write sums to.

Returns

Iterator pointing just beyond the values written to result.

Definition at line 238 of file stl_numeric.h.

4.11.4.251 `template<typename _InputIterator , typename _OutputIterator ,
typename _BinaryOperation > _OutputIterator std::partial_sum (
_InputIterator __first, _InputIterator __last, _OutputIterator
__result, _BinaryOperation __binary_op)`

Return list of partial sums.

Accumulates the values in the range [first,last) using operator+(). As each successive input value is added into the total, that partial sum is written to *result*. Therefore, the first value in result is the first value of the input, the second value in result is the sum of the first and second input values, and so on.

Parameters

first Start of input range.
last End of input range.
result Output to write sums to.

Returns

Iterator pointing just beyond the values written to result.

Definition at line 278 of file stl_numeric.h.

```
4.11.4.252 template<typename _MoneyT > _Put_money<_MoneyT>  
std::put_money ( const _MoneyT & __mon, bool __intl = false )  
[inline]
```

Extended manipulator for inserting money.

Parameters

mon Either long double or a specialization of `basic_string`.
intl A bool indicating whether international format is to be used.

Sent to a stream object, this manipulator inserts *mon*.

Definition at line 294 of file iomanip.

```
4.11.4.253 template<typename _Tp > reference_wrapper<_Tp> std::ref (  
_Tp & __t ) [inline]
```

Denotes a reference should be taken to a variable.

Definition at line 470 of file functional.

Referenced by `ref()`.

```
4.11.4.254 template<typename _Tp > reference_wrapper<_Tp> std::ref (  
reference_wrapper<_Tp> __t ) [inline]
```

Partial specialization.

Definition at line 482 of file functional.

References `ref()`.

4.11.4.255 `template<typename _InputIterator , typename _OutputIterator
, typename _Tp > _OutputIterator std::replace_copy (`
 `_InputIterator __first, _InputIterator __last, _OutputIterator`
 `__result, const _Tp & __old_value, const _Tp & __new_value)`

Copy a sequence, replacing each element of one value with another value.

Parameters

first An input iterator.
last An input iterator.
result An output iterator.
old_value The value to be replaced.
new_value The replacement value.

Returns

The end of the output sequence, `result+(last-first)`.

Copies each element in the input range `[first,last)` to the output range `[result,result+(last-first))` replacing elements equal to `old_value` with `new_value`.

Definition at line 3745 of file `stl_algo.h`.

4.11.4.256 `_Resetiosflags std::resetiosflags (ios_base::fmtflags __mask)`
 `[inline]`

Manipulator for `setf`.

Parameters

mask A format flags mask.

Sent to a stream object, this manipulator resets the specified flags, via `stream.setf(0,mask)`.

Definition at line 65 of file `iomanip`.

4.11.4.257 `template<typename _Tp > void std::return_temporary_buffer (`
 `_Tp* __p) [inline]`

The companion to [get_temporary_buffer\(\)](#).

Parameters

p A buffer previously allocated by `get_temporary_buffer`.

Returns

None.

Frees the memory pointed to by *p*.

Definition at line 113 of file `std_tempbuf.h`.

Referenced by `std::_Temporary_buffer<_ForwardIterator, _Tp>::_Temporary_buffer()`.

4.11.4.258 ios_base& std::right (ios_base & __base) [inline]

Calls `base.setf(ios_base::right, ios_base::adjustfield)`.

Definition at line 926 of file `ios_base.h`.

References `std::ios_base::adjustfield`, `std::ios_base::right`, and `std::ios_base::setf()`.

4.11.4.259 ios_base& std::scientific (ios_base & __base) [inline]

Calls `base.setf(ios_base::scientific, ios_base::floatfield)`.

Definition at line 968 of file `ios_base.h`.

References `std::ios_base::floatfield`, `std::ios_base::scientific`, and `std::ios_base::setf()`.

Referenced by `operator<<()`.

4.11.4.260 new_handler std::set_new_handler (new_handler) throw ()

Takes a replacement handler as the argument, returns the previous handler.

4.11.4.261 _Setbase std::setbase (int __base) [inline]

Manipulator for `setf`.

Parameters

base A numeric base.

Sent to a stream object, this manipulator changes the `ios_base::basefield` flags to `oct`, `dec`, or `hex` when *base* is 8, 10, or 16, accordingly, and to 0 if *base* is any other value.

Definition at line 126 of file `iomanip`.

4.11.4.262 `template<typename _CharT> _Setfill<_CharT> std::setfill (`
`_CharT __c) [inline]`

Manipulator for `fill`.

Parameters

c The new fill character.

Sent to a stream object, this manipulator calls `fill(c)` for that object.

Definition at line 164 of file `iomanip`.

4.11.4.263 `_Setiosflags std::setiosflags (ios_base::fmtflags __mask)`
`[inline]`

Manipulator for `setf`.

Parameters

mask A format flags mask.

Sent to a stream object, this manipulator sets the format flags to *mask*.

Definition at line 95 of file `iomanip`.

4.11.4.264 `_Setprecision std::setprecision (int __n) [inline]`

Manipulator for `precision`.

Parameters

n The new precision.

Sent to a stream object, this manipulator calls `precision(n)` for that object.

Definition at line 194 of file `iomanip`.

4.11.4.265 `_Setw std::setw (int __n) [inline]`

Manipulator for `width`.

Parameters

n The new width.

Sent to a stream object, this manipulator calls `width(n)` for that object.

Definition at line 224 of file `iomanip`.

4.11.4.266 `ios_base& std::showbase (ios_base & __base) [inline]`

Calls `base.setf(ios_base::showbase)`.

Definition at line 813 of file `ios_base.h`.

References `std::ios_base::setf()`, and `std::ios_base::showbase`.

4.11.4.267 `ios_base& std::showpoint (ios_base & __base) [inline]`

Calls `base.setf(ios_base::showpoint)`.

Definition at line 829 of file `ios_base.h`.

References `std::ios_base::setf()`, and `std::ios_base::showpoint`.

4.11.4.268 `ios_base& std::showpos (ios_base & __base) [inline]`

Calls `base.setf(ios_base::showpos)`.

Definition at line 845 of file `ios_base.h`.

References `std::ios_base::setf()`, and `std::ios_base::showpos`.

4.11.4.269 ios_base& std::skipws (ios_base & __base) [inline]

Calls base.setf(ios_base::skipws).

Definition at line 861 of file ios_base.h.

References std::ios_base::setf(), and std::ios_base::skipws.

Referenced by operator>>().

**4.11.4.270 template<typename _Tp , typename _Tp1 , _Lock_policy _Lp>
__shared_ptr<_Tp, _Lp> std::static_pointer_cast (const
__shared_ptr<_Tp1, _Lp > & __r) [inline]**

static_pointer_cast

Definition at line 1119 of file shared_ptr_base.h.

**4.11.4.271 template<typename _Tp , typename _Alloc > void std::swap (
deque<_Tp, _Alloc > & __x, deque<_Tp, _Alloc > & __y)
[inline]**

See [std::deque::swap\(\)](#).

Definition at line 1971 of file stl_deque.h.

References std::deque<_Tp, _Alloc >::swap().

**4.11.4.272 template<typename _Key , typename _Compare , typename _Alloc
> void std::swap (set<_Key, _Compare, _Alloc > & __x, set<
_Key, _Compare, _Alloc > & __y) [inline]**

See [std::set::swap\(\)](#).

Definition at line 748 of file stl_set.h.

References std::set<_Key, _Compare, _Alloc >::swap().

4.11.4.273 `template<typename _Res , typename... _Args> void std::swap (function< _Res(_Args...)> & __x, function< _Res(_Args...)> & __y) [inline]`

Swap the targets of two polymorphic function object wrappers.

This function will not throw an exception.

Definition at line 2250 of file functional.

4.11.4.274 `template<class _T1 , class _T2 > void std::swap (pair< _T1, _T2 > & __x, pair< _T1, _T2 > & __y) [inline]`

See `std::pair::swap()`.

Definition at line 241 of file `stl_pair.h`.

4.11.4.275 `template<typename _CharT , typename _Traits , typename _Alloc > void std::swap (basic_string< _CharT, _Traits, _Alloc > & __lhs, basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline]`

Swap contents of two strings.

Parameters

lhs First string.

rhs Second string.

Exchanges the contents of *lhs* and *rhs* in constant time.

Definition at line 2658 of file `basic_string.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::swap()`.

4.11.4.276 `template<typename _Key , typename _Tp , typename _Compare , typename _Alloc > void std::swap (map< _Key, _Tp, _Compare, _Alloc > & __x, map< _Key, _Tp, _Compare, _Alloc > & __y) [inline]`

See [std::map::swap\(\)](#).

Definition at line 929 of file `std_map.h`.

References `std::map<_Key, _Tp, _Compare, _Alloc >::swap()`.

4.11.4.277 `template<typename _Tp, typename _Alloc > void std::swap (`
`forward_list<_Tp, _Alloc > & __lx, forward_list<_Tp, _Alloc >`
`& __ly) [inline]`

See [std::forward_list::swap\(\)](#).

Definition at line 1296 of file `forward_list.h`.

References `std::forward_list<_Tp, _Alloc >::swap()`.

4.11.4.278 `template<typename _Key, typename _Compare, typename _Alloc`
`> void std::swap (multiset<_Key, _Compare, _Alloc > & __x,`
`multiset<_Key, _Compare, _Alloc > & __y) [inline]`

See [std::multiset::swap\(\)](#).

Definition at line 731 of file `std_multiset.h`.

References `std::multiset<_Key, _Compare, _Alloc >::swap()`.

4.11.4.279 `template<typename _Tp, typename _Alloc > void std::swap (`
`vector<_Tp, _Alloc > & __x, vector<_Tp, _Alloc > & __y)`
`[inline]`

See [std::vector::swap\(\)](#).

Definition at line 1321 of file `std_vector.h`.

References `std::vector<_Tp, _Alloc >::swap()`.

4.11.4.280 `template<typename _Tp, typename _Alloc > void std::swap (list<`
`_Tp, _Alloc > & __x, list<_Tp, _Alloc > & __y) [inline]`

See [std::list::swap\(\)](#).

Definition at line 1624 of file `std_list.h`.

References `std::list<_Tp, _Alloc >::swap()`.

4.11.4.281 `template<typename _Key , typename _Tp , typename _Compare ,
typename _Alloc > void std::swap (multimap< _Key, _Tp,
_Compare, _Alloc > & __x, multimap< _Key, _Tp, _Compare,
_Alloc > & __y) [inline]`

See [std::multimap::swap\(\)](#).

Definition at line 847 of file `stl_multimap.h`.

References `std::multimap< _Key, _Tp, _Compare, _Alloc >::swap()`.

4.11.4.282 `template<typename _CharT > _CharT std::tolower (_CharT __c,
const locale & __loc) [inline]`

Convenience interface to `ctype.tolower(__c)`.

Referenced by `std::regex_traits< _Ch_type >::translate_nocase()`.

4.11.4.283 `template<typename _CharT > _CharT std::toupper (_CharT __c,
const locale & __loc) [inline]`

Convenience interface to `ctype.toupper(__c)`.

4.11.4.284 `template<typename _InputIterator , typename _ForwardIterator >
_ForwardIterator std::uninitialized_copy (_InputIterator __first,
_InputIterator __last, _ForwardIterator __result) [inline]`

Copies the range `[first,last)` into `result`.

Parameters

first An input iterator.

last An input iterator.

result An output iterator.

Returns

`result + (first - last)`

Like `copy()`, but does not require an initialized output range.

Definition at line 109 of file `std_uninitialized.h`.

Referenced by `__gnu_parallel::parallel_sort_mwms_pu()`.

4.11.4.285 `template<typename _InputIterator , typename _Size , typename
_ForwardIterator > _ForwardIterator std::uninitialized_copy_n (
_InputIterator __first, _Size __n, _ForwardIterator __result)
[inline]`

Copies the range `[first,first+n)` into `result`.

Parameters

first An input iterator.
n The number of elements to copy.
result An output iterator.

Returns

`result + n`

Like [copy_n\(\)](#), but does not require an initialized output range.

Definition at line 631 of file `std_uninitialized.h`.

References `__iterator_category()`.

4.11.4.286 `template<typename _ForwardIterator , typename _Tp > void
std::uninitialized_fill (_ForwardIterator __first, _ForwardIterator
__last, const _Tp & __x) [inline]`

Copies the value `x` into the range `[first,last)`.

Parameters

first An input iterator.
last An input iterator.
x The source value.

Returns

Nothing.

Like fill(), but does not require an initialized output range.

Definition at line 166 of file stl_uninitialized.h.

4.11.4.287 `template<typename _ForwardIterator, typename _Size, typename
_Tp> void std::uninitialized_fill_n (_ForwardIterator __first,
_Size __n, const _Tp & __x) [inline]`

Copies the value *x* into the range [first,first+n).

Parameters

first An input iterator.

n The number of copies to make.

x The source value.

Returns

Nothing.

Like fill_n(), but does not require an initialized output range.

Definition at line 220 of file stl_uninitialized.h.

4.11.4.288 `ios_base& std::unitbuf (ios_base & __base) [inline]`

Calls base.setf(ios_base::unitbuf).

Definition at line 893 of file ios_base.h.

References std::ios_base::setf(), and std::ios_base::unitbuf.

4.11.4.289 `ios_base& std::uppercase (ios_base & __base) [inline]`

Calls base.setf(ios_base::uppercase).

Definition at line 877 of file ios_base.h.

References std::ios_base::setf(), and std::ios_base::uppercase.

4.11.4.290 `template<typename _Facet > const _Facet & std::use_facet (const locale & __loc)`

Return a facet.

`use_facet` looks for and returns a reference to a facet of type `Facet` where `Facet` is the template parameter. If `has_facet(locale)` is true, there is a suitable facet to return. It throws [std::bad_cast](#) if the locale doesn't contain a facet of type `Facet`.

Parameters

Facet The facet type to access.

locale The locale to use.

Returns

Reference to facet of type `Facet`.

Exceptions

[std::bad_cast](#) if locale doesn't contain a facet of type `Facet`.

Definition at line 107 of file `locale_classes.tcc`.

Referenced by `std::regex_traits< _Ch_type >::isctype()`, and `std::regex_traits< _Ch_type >::transform()`.

4.11.4.291 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::ws (basic_istream< _CharT, _Traits > & __is)`

Quick and easy way to eat whitespace.

This manipulator extracts whitespace characters, stopping when the next character is non-whitespace, or when the input sequence is empty. If the sequence is empty, `eofbit` is set in the stream, but not `failbit`.

The current locale is used to distinguish whitespace characters.

Example:

```
MyClass mc;

std::cin >> std::ws >> mc;
```

will skip leading whitespace before calling operator<>> on `cin` and your object. Note that the same effect can be achieved by creating a [std::basic_istream::sentry](#) inside your definition of operator<>>.

Definition at line 1018 of file istream.tcc.

References `std::ios_base::eofbit`, `std::ios_base::getloc()`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

4.11.5 Variable Documentation

4.11.5.1 `enable_if<(is_pointer<_Function>::value && is_function<typename remove_pointer<_Function>::type>::value), typename result_of<_Function(_Args...)>::type>::type std::__invoke [inline]`

Invoke a function object, which may be either a member pointer or a function object. The first parameter will tell which.

Definition at line 240 of file functional.

4.11.5.2 `ios_base::Init std::__ioinit [static]`

Linked to standard error (buffered).

Definition at line 75 of file iostream.

4.11.5.3 `ostream std::cerr`

Linked to standard output.

4.11.5.4 `istream std::cin`

Linked to standard input.

4.11.5.5 `ostream std::clog`

Linked to standard error (unbuffered).

4.11.5.6 `ostream std::cout`

Linked to standard input.

4.11.5.7 `constexpr piecewise_construct_t` `std::piecewise_construct`

`piecewise_construct`

Definition at line 75 of file `stl_pair.h`.

4.11.5.8 `wostream` `std::wcerr`

Linked to standard output.

4.11.5.9 `wistream` `std::wcin`

Linked to standard error (buffered).

4.11.5.10 `wostream` `std::wclog`

Linked to standard error (unbuffered).

4.11.5.11 `wostream` `std::wcout`

Linked to standard input.

4.12 `std::__debug` Namespace Reference

GNU debug code, replaces standard behavior with debug behavior.

Classes

- class [bitset](#)

Class `std::bitset` with additional safety/checking/debug instrumentation.

- class `deque`
Class `std::deque` with safety/checking/debug instrumentation.
- class `forward_list`
Class `std::forward_list` with safety/checking/debug instrumentation.
- class `list`
Class `std::list` with safety/checking/debug instrumentation.
- class `map`
Class `std::map` with safety/checking/debug instrumentation.
- class `multimap`
Class `std::multimap` with safety/checking/debug instrumentation.
- class `multiset`
Class `std::multiset` with safety/checking/debug instrumentation.
- class `set`
Class `std::set` with safety/checking/debug instrumentation.
- class `unordered_map`
Class `std::unordered_map` with safety/checking/debug instrumentation.
- class `unordered_multimap`
Class `std::unordered_multimap` with safety/checking/debug instrumentation.
- class `unordered_multiset`
Class `std::unordered_multiset` with safety/checking/debug instrumentation.
- class `unordered_set`
Class `std::unordered_set` with safety/checking/debug instrumentation.
- class `vector`
Class `std::vector` with safety/checking/debug instrumentation.

Functions

- `template<typename _Tp, typename _Alloc >`
`bool operator!= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _-`
`_Alloc > &__rhs)`

- `template<typename _Tp, typename _Alloc >`
`bool operator!= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc >`
`&__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc`
`>`
`bool operator!= (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc`
`> &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__`
`__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`bool operator!= (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x,`
`const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator!= (const set< _Key, _Compare, _Allocator > &__lhs, const set<`
`_Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`bool operator!= (const unordered_multiset< _Value, _Hash, _Pred, _Alloc >`
`&__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator!= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _`
`Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator!= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs,`
`const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator!= (const forward_list< _Tp, _Alloc > &__lx, const forward_`
`list< _Tp, _Alloc > &__ly)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator!= (const multimap< _Key, _Tp, _Compare, _Allocator > &__`
`lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc`
`>`
`bool operator!= (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`
`&__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator!= (const multiset< _Key, _Compare, _Allocator > &__lhs, const`
`multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<size_t _Nb>`
`bitset< _Nb > operator& (const bitset< _Nb > &__x, const bitset< _Nb >`
`&__y)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator< (const set< _Key, _Compare, _Allocator > &__lhs, const set<`
`_Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator< (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _`
`Alloc > &__rhs)`

- `template<typename _Tp, typename _Alloc >`
`bool operator< (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator< (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator< (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator< (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator< (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator< (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`
`std::basic_ostream< _CharT, _Traits > &operator<< (std::basic_ostream< _CharT, _Traits > &__os, const bitset< _Nb > &__x)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator<= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator<= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`
`bool operator<= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator<= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator<= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator<= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator<= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator<= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`

- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator== (const set< _Key, _Compare, _Allocator > &__lhs, const set<`
`_Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`bool operator== (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x,`
`const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`bool operator== (const unordered_multiset< _Value, _Hash, _Pred, _Alloc >`
`&__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator== (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp,`
`_Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator== (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc >`
`&__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator== (const map< _Key, _Tp, _Compare, _Allocator > &__lhs,`
`const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_`
`list< _Tp, _Alloc > &__ly)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc`
`>`
`bool operator== (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _`
`Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc`
`> &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator== (const multimap< _Key, _Tp, _Compare, _Allocator > &__`
`lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc`
`>`
`bool operator== (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`
`&__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator== (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _`
`Alloc > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator== (const multiset< _Key, _Compare, _Allocator > &__lhs, const`
`multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator> (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs,`
`const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator> (const multiset< _Key, _Compare, _Allocator > &__lhs, const`
`multiset< _Key, _Compare, _Allocator > &__rhs)`

- `template<typename _Tp, typename _Alloc >`
`bool operator> (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator> (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator> (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator> (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator> (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator> (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator>= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator>= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator>= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator>= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, bitset< _Nb > &__x)`

- template<size_t _Nb>
bitset< _Nb > **operator**^ (const [bitset](#)< _Nb > &__x, const [bitset](#)< _Nb > &__y)
- template<size_t _Nb>
bitset< _Nb > **operator**| (const [bitset](#)< _Nb > &__x, const [bitset](#)< _Nb > &__y)
- template<typename _Tp, typename _Alloc >
void **swap** ([vector](#)< _Tp, _Alloc > &__lhs, [vector](#)< _Tp, _Alloc > &__rhs)
- template<typename _Tp, typename _Alloc >
void **swap** ([forward_list](#)< _Tp, _Alloc > &__lx, [forward_list](#)< _Tp, _Alloc > &__ly)
- template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >
void **swap** ([unordered_multiset](#)< _Value, _Hash, _Pred, _Alloc > &__x, [unordered_multiset](#)< _Value, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
void **swap** ([multimap](#)< _Key, _Tp, _Compare, _Allocator > &__lhs, [multimap](#)< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >
void **swap** ([unordered_map](#)< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, [unordered_map](#)< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key, typename _Compare, typename _Allocator >
void **swap** ([set](#)< _Key, _Compare, _Allocator > &__x, [set](#)< _Key, _Compare, _Allocator > &__y)
- template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >
void **swap** ([unordered_set](#)< _Value, _Hash, _Pred, _Alloc > &__x, [unordered_set](#)< _Value, _Hash, _Pred, _Alloc > &__y)
- template<typename _Tp, typename _Alloc >
void **swap** ([deque](#)< _Tp, _Alloc > &__lhs, [deque](#)< _Tp, _Alloc > &__rhs)
- template<typename _Key, typename _Compare, typename _Allocator >
void **swap** ([multiset](#)< _Key, _Compare, _Allocator > &__x, [multiset](#)< _Key, _Compare, _Allocator > &__y)
- template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >
void **swap** ([unordered_multimap](#)< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, [unordered_multimap](#)< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
void **swap** ([map](#)< _Key, _Tp, _Compare, _Allocator > &__lhs, [map](#)< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Tp, typename _Alloc >
void **swap** ([list](#)< _Tp, _Alloc > &__lhs, [list](#)< _Tp, _Alloc > &__rhs)

4.12.1 Detailed Description

GNU debug code, replaces standard behavior with debug behavior. Macros and namespaces used by the implementation outside of debug wrappers to verify certain properties. The `__glibcxx_requires_XXX` macros are merely wrappers around the `__glibcxx_check_XXX` wrappers when we are compiling with debug mode, but disappear when we are in release mode so that there is no checking performed in, e.g., the standard library algorithms.

4.12.2 Function Documentation

4.12.2.1 `template<typename _Tp, typename _Alloc> bool
std::__debug::operator<= (const forward_list< _Tp, _Alloc> &
__lx, const forward_list< _Tp, _Alloc> & __ly) [inline]`

Based on operator<.

Definition at line 712 of file debug/forward_list.

4.12.2.2 `template<typename _Tp, typename _Alloc> bool
std::__debug::operator> (const forward_list< _Tp, _Alloc> & __lx,
const forward_list< _Tp, _Alloc> & __ly) [inline]`

Based on operator<.

Definition at line 698 of file debug/forward_list.

4.12.2.3 `template<typename _Tp, typename _Alloc> bool
std::__debug::operator>= (const forward_list< _Tp, _Alloc> &
__lx, const forward_list< _Tp, _Alloc> & __ly) [inline]`

Based on operator<.

Definition at line 705 of file debug/forward_list.

4.12.2.4 `template<typename _Tp, typename _Alloc> void std::__debug::swap
(forward_list< _Tp, _Alloc> & __lx, forward_list< _Tp, _Alloc>
& __ly) [inline]`

See [std::forward_list::swap\(\)](#).

Definition at line 719 of file `debug/forward_list`.

4.13 std::__detail Namespace Reference

Implementation details not part of the namespace `std` interface.

Classes

- struct [_List_node_base](#)
Common part of a node in the list.

Functions

- `template<class _Iterator >`
`std::iterator_traits< _Iterator >::difference_type __distance_fw (_Iterator __first, _Iterator __last, std::input_iterator_tag)`
- `template<class _Iterator >`
`std::iterator_traits< _Iterator >::difference_type __distance_fw (_Iterator __first, _Iterator __last, std::forward_iterator_tag)`
- `template<class _Iterator >`
`std::iterator_traits< _Iterator >::difference_type __distance_fw (_Iterator __first, _Iterator __last)`
- `template<typename _InputIterator , typename _OutputIterator , typename _UnaryOperation >`
`_OutputIterator __transform (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _Value , bool __cache>`
`bool operator!= (const _Node_iterator_base< _Value, __cache > &__x, const _Node_iterator_base< _Value, __cache > &__y)`
- `template<typename _Value , bool __cache>`
`bool operator!= (const _Hashtable_iterator_base< _Value, __cache > &__x, const _Hashtable_iterator_base< _Value, __cache > &__y)`
- `template<typename _Value , bool __cache>`
`bool operator== (const _Node_iterator_base< _Value, __cache > &__x, const _Node_iterator_base< _Value, __cache > &__y)`
- `template<typename _Value , bool __cache>`
`bool operator== (const _Hashtable_iterator_base< _Value, __cache > &__x, const _Hashtable_iterator_base< _Value, __cache > &__y)`

Variables

- `const unsigned long __prime_list []`

4.13.1 Detailed Description

Implementation details not part of the namespace `std` interface.

4.14 `std::__parallel` Namespace Reference

GNU parallel code, replaces standard behavior with parallel behavior.

Classes

- struct [`_CRandNumber`](#)
Functor wrapper for `std::rand()`.

Functions

- `template<typename _Iter, typename _Tp, typename _IteratorTag >`
`_Tp __accumulate_switch (_Iter __begin, _Iter __end, _Tp __init, _-`
`IteratorTag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation, typename _IteratorTag >`
`_Tp __accumulate_switch (_Iter __begin, _Iter __end, _Tp __init, _-`
`BinaryOperation __binary_op, _IteratorTag)`
- `template<typename _RAIter, typename _Tp, typename _BinaryOperation >`
`_Tp __accumulate_switch (_RAIter __begin, _RAIter __end, _Tp __init, _-`
`BinaryOperation __binary_op, random_access_iterator_tag, __gnu_parallel::_-`
`Parallelism __parallelism_tag=__gnu_parallel::parallel_unbalanced)`
- `template<typename _Iter, typename _Tp, typename _Tag >`
`_Tp __accumulate_switch (_Iter, _Iter, _Tp, _Tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper, typename _Tag >`
`_Tp __accumulate_switch (_Iter, _Iter, _Tp, _BinaryOper, _Tag)`
- `template<typename _RAIter, typename _Tp, typename _BinaryOper >`
`_Tp __accumulate_switch (_RAIter, _RAIter, _Tp, _BinaryOper, random_-`
`access_iterator_tag, __gnu_parallel::_Parallelism __parallelism=__gnu_-`
`parallel::parallel_unbalanced)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper, typename _Tag1, typename`
`_Tag2 >`
`_OIter __adjacent_difference_switch (_Iter, _Iter, _OIter, _BinaryOper, _-`
`Tag1, _Tag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`
`_OIter __adjacent_difference_switch (_Iter, _Iter, _OIter, _BinaryOper,`
`random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_-`
`Parallelism __parallelism=__gnu_parallel::parallel_unbalanced)`

- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation, typename _IteratorTag1, typename _IteratorTag2 >`
`_OutputIterator __adjacent_difference_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator __adjacent_difference_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _RAIter >`
`_RAIter __adjacent_find_switch (_RAIter __begin, _RAIter __end, random_access_iterator_tag)`
- `template<typename _FIterator, typename _IteratorTag >`
`_FIterator __adjacent_find_switch (_FIterator __begin, _FIterator __end, _IteratorTag)`
- `template<typename _FIterator, typename _BinaryPredicate, typename _IteratorTag >`
`_FIterator __adjacent_find_switch (_FIterator __begin, _FIterator __end, _BinaryPredicate __pred, _IteratorTag)`
- `template<typename _RAIter, typename _BinaryPredicate >`
`_RAIter __adjacent_find_switch (_RAIter __begin, _RAIter __end, _BinaryPredicate __pred, random_access_iterator_tag)`
- `template<typename _FIter, typename _IterTag >`
`_FIter __adjacent_find_switch (_FIter, _FIter, _IterTag)`
- `template<typename _FIter, typename _BiPredicate, typename _IterTag >`
`_FIter __adjacent_find_switch (_FIter, _FIter, _BiPredicate, _IterTag)`
- `template<typename _RAIter, typename _BiPredicate >`
`_RAIter __adjacent_find_switch (_RAIter, _RAIter, _BiPredicate, random_access_iterator_tag)`
- `template<typename _RAIter, typename _Predicate >`
`iterator_traits< _RAIter >::difference_type __count_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag=__gnu_parallel::parallel_unbalanced)`
- `template<typename _Iter, typename _Predicate, typename _IteratorTag >`
`iterator_traits< _Iter >::difference_type __count_if_switch (_Iter __begin, _Iter __end, _Predicate __pred, _IteratorTag)`
- `template<typename _Iter, typename _Predicate, typename _IterTag >`
`iterator_traits< _Iter >::difference_type __count_if_switch (_Iter, _Iter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Tp >`
`iterator_traits< _RAIter >::difference_type __count_switch (_RAIter __begin, _RAIter __end, const _Tp &__value, random_access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag=__gnu_parallel::parallel_unbalanced)`
- `template<typename _Iter, typename _Tp, typename _IteratorTag >`
`iterator_traits< _Iter >::difference_type __count_switch (_Iter __begin, _Iter __end, const _Tp &__value, _IteratorTag)`

- `template<typename _Iter, typename _Tp, typename _IterTag >`
`iterator_traits< _Iter >::difference_type __count_switch (_Iter, _Iter, const`
`_Tp &, _IterTag)`
- `template<typename _Iter, typename _FIterator, typename _IteratorTag1, typename _IteratorTag2`
`>`
`_Iter __find_first_of_switch (_Iter __begin1, _Iter __end1, _FIterator __`
`begin2, _FIterator __end2, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter, typename _FIterator, typename _BinaryPredicate, typename _`
`IteratorTag >`
`_RAIter __find_first_of_switch (_RAIter __begin1, _RAIter __end1, _FIterator`
`__begin2, _FIterator __end2, _BinaryPredicate __comp, random_access_`
`iterator_tag, _IteratorTag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate, typename _`
`IteratorTag1, typename _IteratorTag2 >`
`_Iter __find_first_of_switch (_Iter __begin1, _Iter __end1, _FIterator _`
`begin2, _FIterator __end2, _BinaryPredicate __comp, _IteratorTag1, _`
`IteratorTag2)`
- `template<typename _Iter, typename _Filter, typename _IterTag1, typename _IterTag2 >`
`_Iter __find_first_of_switch (_Iter, _Iter, _Filter, _Filter, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _Filter, typename _BiPredicate, typename _IterTag >`
`_RAIter __find_first_of_switch (_RAIter, _RAIter, _Filter, _Filter, _BiPredicate,`
`random_access_iterator_tag, _IterTag)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate, typename _IterTag1, type-`
`name _IterTag2 >`
`_Iter __find_first_of_switch (_Iter, _Iter, _Filter, _Filter, _BiPredicate, _`
`IterTag1, _IterTag2)`
- `template<typename _Iter, typename _Predicate, typename _IteratorTag >`
`_Iter __find_if_switch (_Iter __begin, _Iter __end, _Predicate __pred, _`
`IteratorTag)`
- `template<typename _RAIter, typename _Predicate >`
`_RAIter __find_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred,`
`random_access_iterator_tag)`
- `template<typename _Iter, typename _Predicate, typename _IterTag >`
`_Iter __find_if_switch (_Iter, _Iter, _Predicate, _IterTag)`
- `template<typename _Iter, typename _Tp, typename _IteratorTag >`
`_Iter __find_switch (_Iter __begin, _Iter __end, const _Tp &__val, _`
`IteratorTag)`
- `template<typename _RAIter, typename _Tp >`
`_RAIter __find_switch (_RAIter __begin, _RAIter __end, const _Tp &__val,`
`random_access_iterator_tag)`
- `template<typename _Iter, typename _Tp, typename _IterTag >`
`_Iter __find_switch (_Iter, _Iter, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Function >`
`_Function __for_each_switch (_RAIter __begin, _RAIter __end, _Function`

- [__f](#), [random_access_iterator_tag](#), [__gnu_parallel::__Parallelism](#) [__parallelism_tag=__gnu_parallel::parallel_balanced](#))
- template<typename [_Iter](#) , typename [_Function](#) , typename [_IteratorTag](#) >
[_Function](#) **__for_each_switch** ([_Iter](#) [__begin](#), [_Iter](#) [__end](#), [_Function](#) [__f](#), [_IteratorTag](#))
- template<typename [_Iter](#) , typename [_Function](#) , typename [_IterTag](#) >
[_Function](#) **__for_each_switch** ([_Iter](#), [_Iter](#), [_Function](#), [_IterTag](#))
- template<typename [_OutputIterator](#) , typename [_Size](#) , typename [_Generator](#) , typename [_IteratorTag](#) >
[_OutputIterator](#) **__generate_n_switch** ([_OutputIterator](#) [__begin](#), [_Size](#) [__n](#), [_Generator](#) [__gen](#), [_IteratorTag](#))
- template<typename [_RAIter](#) , typename [_Size](#) , typename [_Generator](#) >
[_RAIter](#) **__generate_n_switch** ([_RAIter](#) [__begin](#), [_Size](#) [__n](#), [_Generator](#) [__gen](#), [random_access_iterator_tag](#), [__gnu_parallel::__Parallelism](#) [__parallelism_tag=__gnu_parallel::parallel_balanced](#))
- template<typename [_OIter](#) , typename [_Size](#) , typename [_Generator](#) , typename [_IterTag](#) >
[_OIter](#) **__generate_n_switch** ([_OIter](#), [_Size](#), [_Generator](#), [_IterTag](#))
- template<typename [_FIterator](#) , typename [_Generator](#) , typename [_IteratorTag](#) >
void **__generate_switch** ([_FIterator](#) [__begin](#), [_FIterator](#) [__end](#), [_Generator](#) [__gen](#), [_IteratorTag](#))
- template<typename [_RAIter](#) , typename [_Generator](#) >
void **__generate_switch** ([_RAIter](#) [__begin](#), [_RAIter](#) [__end](#), [_Generator](#) [__gen](#), [random_access_iterator_tag](#), [__gnu_parallel::__Parallelism](#) [__parallelism_tag=__gnu_parallel::parallel_balanced](#))
- template<typename [_FIter](#) , typename [_Generator](#) , typename [_IterTag](#) >
void **__generate_switch** ([_FIter](#), [_FIter](#), [_Generator](#), [_IterTag](#))
- template<typename [_RAIter1](#) , typename [_RAIter2](#) , typename [_Tp](#) , typename [BinaryFunction1](#) , typename [BinaryFunction2](#) >
[_Tp](#) **__inner_product_switch** ([_RAIter1](#), [_RAIter1](#), [_RAIter2](#), [_Tp](#), [BinaryFunction1](#), [BinaryFunction2](#), [random_access_iterator_tag](#), [random_access_iterator_tag](#), [__gnu_parallel::__Parallelism](#) [__parallelism_tag=__gnu_parallel::parallel_unbalanced](#))
- template<typename [_Iter1](#) , typename [_Iter2](#) , typename [_Tp](#) , typename [BinaryFunction1](#) , typename [BinaryFunction2](#) , typename [_Tag1](#) , typename [_Tag2](#) >
[_Tp](#) **__inner_product_switch** ([_Iter1](#), [_Iter1](#), [_Iter2](#), [_Tp](#), [BinaryFunction1](#), [BinaryFunction2](#), [_Tag1](#), [_Tag2](#))
- template<typename [_RAIter1](#) , typename [_RAIter2](#) , typename [_Tp](#) , typename [BinaryFunction1](#) , typename [BinaryFunction2](#) >
[_Tp](#) **__inner_product_switch** ([_RAIter1](#) [__first1](#), [_RAIter1](#) [__last1](#), [_RAIter2](#) [__first2](#), [_Tp](#) [__init](#), [BinaryFunction1](#) [__binary_op1](#), [BinaryFunction2](#) [__binary_op2](#), [random_access_iterator_tag](#), [random_access_iterator_tag](#), [__gnu_parallel::__Parallelism](#) [__parallelism_tag=__gnu_parallel::parallel_unbalanced](#))
- template<typename [_Iter1](#) , typename [_Iter2](#) , typename [_Tp](#) , typename [BinaryFunction1](#) , typename [BinaryFunction2](#) , typename [_IteratorTag1](#) , typename [_IteratorTag2](#) >
[_Tp](#) **__inner_product_switch** ([_Iter1](#) [__first1](#), [_Iter1](#) [__last1](#), [_Iter2](#) [__first2](#),

- ```

_Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2,
_IteratorTag1, _IteratorTag2)

```
- ```

template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1,
typename _IteratorTag2 >
bool __lexicographical_compare_switch (_Iter1 __begin1, _Iter1 __end1,
_Iter2 __begin2, _Iter2 __end2, _Predicate __pred, _IteratorTag1, _
IteratorTag2)

```
 - ```

template<typename _RAIter1, typename _RAIter2, typename _Predicate >
bool __lexicographical_compare_switch (_RAIter1 __begin1, _RAIter1 _
_end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random_
access_iterator_tag, random_access_iterator_tag)

```
  - ```

template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IterTag1, type-
name _IterTag2 >
bool __lexicographical_compare_switch (_Iter1, _Iter1, _Iter2, _Iter2, _
Predicate, _IterTag1, _IterTag2)

```
 - ```

template<typename _Filter, typename _Compare, typename _IterTag >
_Filter __max_element_switch (_Filter, _Filter, _Compare, _IterTag)

```
  - ```

template<typename _Filterator, typename _Compare, typename _IteratorTag >
_Filterator __max_element_switch (_Filterator __begin, _Filterator __end, _
Compare __comp, _IteratorTag)

```
 - ```

template<typename _RAIter, typename _Compare >
_RAIter __max_element_switch (_RAIter __begin, _RAIter __end, _
Compare __comp, random_access_iterator_tag, gnu_parallel::Parallelism _
_parallelism_tag=gnu_parallel::parallel_balanced)

```
  - ```

template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare, typename
_IterTag1, typename _IterTag2, typename _IterTag3 >
_OIter __merge_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, _
IterTag1, _IterTag2, _IterTag3)

```
 - ```

template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >
_OIter __merge_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _
Compare, random_access_iterator_tag, random_access_iterator_tag, random_
access_iterator_tag)

```
  - ```

template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare,
typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >
_OutputIterator __merge_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 _
__begin2, _Iter2 __end2, _OutputIterator __result, _Compare __comp, _
IteratorTag1, _IteratorTag2, _IteratorTag3)

```
 - ```

template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare >
_OutputIterator __merge_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 _
__begin2, _Iter2 __end2, _OutputIterator __result, _Compare __comp, random_
access_iterator_tag, random_access_iterator_tag, random_
access_iterator_tag)

```
  - ```

template<typename _Filter, typename _Compare, typename _IterTag >
_Filter __min_element_switch (_Filter, _Filter, _Compare, _IterTag)

```
 - ```

template<typename _Filterator, typename _Compare, typename _IteratorTag >
_Filterator __min_element_switch (_Filterator __begin, _Filterator __end, _
Compare __comp, _IteratorTag)

```

- `template<typename _RAIter, typename _Compare >`  
`_RAIter __min_element_switch (_RAIter __begin, _RAIter __end, _-`  
`Compare __comp, random\_access\_iterator\_tag, \_\_gnu\_parallel::\_\_Parallelism _-`  
`__parallelism_tag=\_\_gnu\_parallel::parallel\_balanced)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1,`  
`typename _IteratorTag2 >`  
`pair< _Iter1, _Iter2 > __mismatch_switch (_Iter1 __begin1, _Iter1 __end1,`  
`_Iter2 __begin2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`  
`pair< _RAIter1, _RAIter2 > __mismatch_switch (_RAIter1 __begin1, _-`  
`RAIter1 __end1, _RAIter2 __begin2, _Predicate __pred, random\_access\_-`  
`iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IterTag1, type-`  
`name _IterTag2 >`  
`pair< _Iter1, _Iter2 > __mismatch_switch (_Iter1, _Iter1, _Iter2, _-`  
`Predicate, _IterTag1, _IterTag2)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator __partial_sum_switch (_Iter __begin, _Iter __end, _-`  
`OutputIterator __result, _BinaryOperation __bin_op, random\_access\_iterator\_-`  
`tag, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation, typename`  
`_IteratorTag1, typename _IteratorTag2 >`  
`_OutputIterator __partial_sum_switch (_Iter __begin, _Iter __end, _-`  
`OutputIterator __result, _BinaryOperation __bin_op, _IteratorTag1, _-`  
`IteratorTag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper, typename _Tag1, typename`  
`_Tag2 >`  
`_OIter __partial_sum_switch (_Iter, _Iter, _OIter, _BinaryOper, _Tag1, _-`  
`Tag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`  
`_OIter __partial_sum_switch (_Iter, _Iter, _OIter, _BinaryOper, random\_-`  
`access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _FIterator, typename _Predicate, typename _IteratorTag >`  
`_FIterator __partition_switch (_FIterator __begin, _FIterator __end, _Predicate`  
`__pred, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate >`  
`_RAIter __partition_switch (_RAIter __begin, _RAIter __end, _Predicate __-`  
`pred, random\_access\_iterator\_tag)`
- `template<typename _FIter, typename _Predicate, typename _IterTag >`  
`_FIter __partition_switch (_FIter, _FIter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Predicate, typename _Tp >`  
`void __replace_if_switch (_RAIter __begin, _RAIter __end, _Predicate __-`  
`pred, const _Tp &__new_value, random\_access\_iterator\_tag, \_\_gnu\_parallel::\_\_-`  
`Parallelism __parallelism_tag=\_\_gnu\_parallel::parallel\_balanced)`

- `template<typename _FIterator, typename _Predicate, typename _Tp, typename _IteratorTag >`  
`void __replace_if_switch (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value, _IteratorTag)`
- `template<typename _FIter, typename _Predicate, typename _Tp, typename _IterTag >`  
`void __replace_if_switch (_FIter, _FIter, _Predicate, const _Tp &, _IterTag)`
- `template<typename _FIter, typename _Tp, typename _IterTag >`  
`void __replace_switch (_FIter, _FIter, const _Tp &, const _Tp &, _IterTag)`
- `template<typename _FIterator, typename _Tp, typename _IteratorTag >`  
`void __replace_switch (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value, _IteratorTag)`
- `template<typename _RAIter, typename _Tp >`  
`void __replace_switch (_RAIter __begin, _RAIter __end, const _Tp &__old_value, const _Tp &__new_value, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag=gnu\_parallel::parallel\_balanced)`
- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BiPredicate >`  
`_RAIter __search_n_switch (_RAIter, _RAIter, _Integer, const _Tp &, _BiPredicate, random\_access\_iterator\_tag)`
- `template<typename _FIter, typename _Integer, typename _Tp, typename _BiPredicate, typename _IterTag >`  
`_FIter __search_n_switch (_FIter, _FIter, _Integer, const _Tp &, _BiPredicate, _IterTag)`
- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BinaryPredicate >`  
`_RAIter __search_n_switch (_RAIter __begin, _RAIter __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, random\_access\_iterator\_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate, typename _IteratorTag >`  
`_FIterator __search_n_switch (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, _IteratorTag)`
- `template<typename _FIter1, typename _FIter2, typename _BiPredicate, typename _IterTag1, typename _IterTag2 >`  
`_FIter1 __search_switch (_FIter1, _FIter1, _FIter2, _FIter2, _BiPredicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter1, typename _RAIter2 >`  
`_RAIter1 __search_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _FIterator1, typename _FIterator2, typename _IteratorTag1, typename _IteratorTag2 >`  
`_FIterator1 __search_switch (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _BinaryPredicate >`  
`_RAIter1 __search_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _BinaryPredicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`

- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2 >`  
`_FIterator1 __search_switch (_FIterator1 __begin1, _FIterator1 __end1,`  
`_FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred, _`  
`IteratorTag1, _IteratorTag2)`
- `template<typename _FIter1, typename _FIter2, typename _IterTag1, typename _IterTag2 >`  
`_FIter1 __search_switch (_FIter1, _FIter1, _FIter2, _FIter2, _IterTag1, _`  
`IterTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _BiPredicate >`  
`_RAIter1 __search_switch (_RAIter1, _RAIter1, _RAIter2, _RAIter2, _`  
`BiPredicate, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator,`  
`typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`  
`_OutputIterator __set_difference_switch (_IIter1 __begin1, _IIter1 __end1, _`  
`IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _`  
`IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _`  
`Predicate >`  
`_Output_RAIter __set_difference_switch (_RAIter1 __begin1, _RAIter1 _`  
`__end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _`  
`Predicate __pred, random_access_iterator_tag, random_access_iterator_tag,`  
`random_access_iterator_tag)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OIter, typename`  
`_IterTag1, typename _IterTag2, typename _IterTag3 >`  
`_OIter __set_difference_switch (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _`  
`Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator,`  
`typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`  
`_OutputIterator __set_intersection_switch (_IIter1 __begin1, _IIter1 __end1,`  
`_IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred,`  
`_IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _`  
`Predicate >`  
`_Output_RAIter __set_intersection_switch (_RAIter1 __begin1, _RAIter1 _`  
`__end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _`  
`Predicate __pred, random_access_iterator_tag, random_access_iterator_tag,`  
`random_access_iterator_tag)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OIter, typename`  
`_IterTag1, typename _IterTag2, typename _IterTag3 >`  
`_OIter __set_intersection_switch (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _`  
`Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _`  
`Predicate >`  
`_Output_RAIter __set_symmetric_difference_switch (_RAIter1 __begin1, _`  
`RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter`

[\\_\\_result](#), [\\_Predicate](#) [\\_\\_pred](#), [random\\_access\\_iterator\\_tag](#), [random\\_access\\_iterator\\_tag](#))

- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`  
`_OutputIterator` **\_\_set\_symmetric\_difference\_switch** (`_Iter1` `__begin1`, `_Iter1` `__end1`, `_Iter2` `__begin2`, `_Iter2` `__end2`, `_OutputIterator` `__result`, `_Predicate` `__pred`, `_IteratorTag1`, `_IteratorTag2`, `_IteratorTag3`)
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3 >`  
`_OIter` **\_\_set\_symmetric\_difference\_switch** (`_Iter1`, `_Iter1`, `_Iter2`, `_Iter2`, `_OIter`, `_Predicate`, `_IterTag1`, `_IterTag2`, `_IterTag3`)
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3 >`  
`_OIter` **\_\_set\_union\_switch** (`_Iter1`, `_Iter1`, `_Iter2`, `_Iter2`, `_OIter`, `_Predicate`, `_IterTag1`, `_IterTag2`, `_IterTag3`)
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`  
`_OutputIterator` **\_\_set\_union\_switch** (`_Iter1` `__begin1`, `_Iter1` `__end1`, `_Iter2` `__begin2`, `_Iter2` `__end2`, `_OutputIterator` `__result`, `_Predicate` `__pred`, `_IteratorTag1`, `_IteratorTag2`, `_IteratorTag3`)
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate >`  
`_Output_RAIter` **\_\_set\_union\_switch** (`_RAIter1` `__begin1`, `_RAIter1` `__end1`, `_RAIter2` `__begin2`, `_RAIter2` `__end2`, `_Output_RAIter` `__result`, `_Predicate` `__pred`, [random\\_access\\_iterator\\_tag](#), [random\\_access\\_iterator\\_tag](#), [random\\_access\\_iterator\\_tag](#))
- `template<typename _Iter, typename _OIter, typename _UnaryOperation, typename _IterTag1, typename _IterTag2 >`  
`_OIter` **\_\_transform1\_switch** (`_Iter`, `_Iter`, `_OIter`, `_UnaryOperation`, `_IterTag1`, `_IterTag2`)
- `template<typename _RAIter, typename _RAOIter, typename _UnaryOperation >`  
`_RAOIter` **\_\_transform1\_switch** (`_RAIter`, `_RAIter`, `_RAOIter`, `_UnaryOperation`, [random\\_access\\_iterator\\_tag](#), [random\\_access\\_iterator\\_tag](#), [\\_\\_gnu\\_parallel::\\_\\_Parallelism](#) `__parallelism=__gnu_parallel::parallel_balanced`)
- `template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation >`  
`_RAIter2` **\_\_transform1\_switch** (`_RAIter1` `__begin`, `_RAIter1` `__end`, `_RAIter2` `__result`, `_UnaryOperation` `__unary_op`, [random\\_access\\_iterator\\_tag](#), [random\\_access\\_iterator\\_tag](#), [\\_\\_gnu\\_parallel::\\_\\_Parallelism](#) `__parallelism_tag=__gnu_parallel::parallel_balanced`)
- `template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation, typename _IteratorTag1, typename _IteratorTag2 >`  
`_RAIter2` **\_\_transform1\_switch** (`_RAIter1` `__begin`, `_RAIter1` `__end`, `_RAIter2` `__result`, `_UnaryOperation` `__unary_op`, `_IteratorTag1`, `_IteratorTag2`)
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BiOperation >`



- `_RAIter3 __transform2_switch (_RAIter1, _RAIter1, _RAIter2, _RAIter3, _BiOperation, random\_access\_iterator\_tag, random\_access\_iterator\_tag, random\_access\_iterator\_tag, \_\_gnu\_parallel::Parallelism __parallelism=__gnu_parallel::parallel_balanced)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation, typename _Tag1, typename _Tag2, typename _Tag3 >  
_OIter __transform2_switch (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, _Tag1, _Tag2, _Tag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BinaryOperation >  
_RAIter3 __transform2_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter3 __result, _BinaryOperation __binary_op, random\_access\_iterator\_tag, random\_access\_iterator\_tag, random\_access\_iterator\_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation, typename _Tag1, typename _Tag2, typename _Tag3 >  
_OutputIterator __transform2_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, _Tag1, _Tag2, _Tag3)`
- `template<typename _Iter, typename _OIter, typename _Predicate, typename _IterTag1, typename _IterTag2 >  
_OIter __unique_copy_switch (_Iter, _Iter, _OIter, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _RandomAccess_OIter, typename _Predicate >  
_RandomAccess_OIter __unique_copy_switch (_RAIter, _RAIter, _RandomAccess_OIter, _Predicate, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >  
_OutputIterator __unique_copy_switch (_Iter __begin, _Iter __last, _OutputIterator __out, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter, typename RandomAccessOutputIterator, typename _Predicate >  
RandomAccessOutputIterator __unique_copy_switch (_RAIter __begin, _RAIter __last, RandomAccessOutputIterator __out, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation >  
_Tp __accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation >  
_Tp __accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation >  
_Tp __accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op)`

- `template<typename _Iter, typename _Tp, typename _BinaryOper >`  
`_Tp accumulate (_Iter, _Iter, _Tp, _BinaryOper)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper >`  
`_Tp accumulate (_Iter, _Iter, _Tp, _BinaryOper, \_\_gnu\_parallel::\_\_Parallelism)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper >`  
`_Tp accumulate (_Iter, _Iter, _Tp, _BinaryOper, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp >`  
`_Tp accumulate (_Iter __begin, _Iter __end, _Tp __init, \_\_gnu\_parallel::\_\_Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp >`  
`_Tp accumulate (_Iter __begin, _Iter __end, _Tp __init, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp >`  
`_Tp accumulate (_Iter __begin, _Iter __end, _Tp __init)`
- `template<typename _Iter, typename _OIter >`  
`_OIter adjacent_difference (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`  
`_OIter adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper)`
- `template<typename _Iter, typename _OIter >`  
`_OIter adjacent_difference (_Iter, _Iter, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter >`  
`_OIter adjacent_difference (_Iter, _Iter, _OIter, \_\_gnu\_parallel::\_\_Parallelism)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`  
`_OIter adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper, \_\_gnu\_parallel::\_\_Parallelism)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, \_\_gnu\_parallel::\_\_Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`  
`_OIter adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result)`

- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _`  
`OutputIterator __result, _BinaryOperation __binary_op, \_\_gnu\_parallel::`  
`Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _`  
`OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _FIterator >`  
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end, \_\_gnu-`  
`parallel::sequential\_tag)`
- `template<typename _FIterator, typename _BinaryPredicate >`  
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end, _`  
`BinaryPredicate __binary_pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator >`  
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator, typename _BinaryPredicate >`  
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end, _`  
`BinaryPredicate __pred)`
- `template<typename _FIter >`  
`_FIter adjacent_find (_FIter, _FIter)`
- `template<typename _FIter >`  
`_FIter adjacent_find (_FIter, _FIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIter, typename _BiPredicate >`  
`_FIter adjacent_find (_FIter, _FIter, _BiPredicate)`
- `template<typename _FIter, typename _BiPredicate >`  
`_FIter adjacent_find (_FIter, _FIter, _BiPredicate, \_\_gnu\_parallel::sequential-`  
`tag)`
- `template<typename _Iter, typename _Tp >`  
`iterator_traits< _Iter >::difference_type count (_Iter __begin, _Iter __end,`  
`const _Tp &__value, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp >`  
`iterator_traits< _Iter >::difference_type count (_Iter __begin, _Iter __end,`  
`const _Tp &__value, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp >`  
`iterator_traits< _Iter >::difference_type count (_Iter __begin, _Iter __end,`  
`const _Tp &__value)`
- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >::difference_type count_if (_Iter __begin, _Iter __end,`  
`_Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >::difference_type count_if (_Iter __begin, _Iter __end,`  
`_Predicate __pred, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >::difference_type count_if (_Iter __begin, _Iter __end,`  
`_Predicate __pred)`

- `template<typename _Iter1, typename _Iter2 >`  
`bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`
- `template<typename _Iter, typename _Tp >`  
`_Iter find (_Iter __begin, _Iter __end, const _Tp &__val, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp >`  
`_Iter find (_Iter __begin, _Iter __end, const _Tp &__val)`
- `template<typename _Iter, typename _FIterator >`  
`_Iter find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate >`  
`_Iter find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate >`  
`_Iter find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp)`
- `template<typename _Iter, typename _FIterator >`  
`_Iter find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2)`
- `template<typename _Iter, typename _Filter >`  
`_Iter find_first_of (_Iter, _Iter, _Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate >`  
`_Iter find_first_of (_Iter, _Iter, _Filter, _Filter, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate >`  
`_Iter find_first_of (_Iter, _Iter, _Filter, _Filter, _BiPredicate)`
- `template<typename _Iter, typename _Filter >`  
`_Iter find_first_of (_Iter, _Iter, _Filter, _Filter)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter find_if (_Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter find_if (_Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _Iter, typename _Function >`  
`_Function for_each (_Iter __begin, _Iter __end, _Function __f, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iterator, typename _Function >`  
`_Function for_each (_Iterator __begin, _Iterator __end, _Function __f, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iterator, typename _Function >`  
`_Function for_each (_Iterator __begin, _Iterator __end, _Function __f)`
- `template<typename _Iter, typename _Function >`  
`_Function for_each (_Iter, _Iter, _Function)`
- `template<typename _FIterator, typename _Generator >`  
`void generate (_FIterator __begin, _FIterator __end, _Generator __gen, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Generator >`  
`void generate (_FIterator __begin, _FIterator __end, _Generator __gen, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Generator >`  
`void generate (_FIterator __begin, _FIterator __end, _Generator __gen)`
- `template<typename _FIter, typename _Generator >`  
`void generate (_FIter, _FIter, _Generator)`
- `template<typename _FIter, typename _Generator >`  
`void generate (_FIter, _FIter, _Generator, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIter, typename _Generator >`  
`void generate (_FIter, _FIter, _Generator, \_\_gnu\_parallel::Parallelism)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`_OutputIterator generate_n (_OutputIterator __begin, _Size __n, _Generator __gen, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`_OutputIterator generate_n (_OutputIterator __begin, _Size __n, _Generator __gen, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`_OutputIterator generate_n (_OutputIterator __begin, _Size __n, _Generator __gen)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter generate_n (_OIter, _Size, _Generator)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter generate_n (_OIter, _Size, _Generator, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter generate_n (_OIter, _Size, _Generator, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`  
`_Tp inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`  
`_Tp inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init)`

- `template<typename _Iter1, typename _Iter2, typename _Tp >`  
`_Tp inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init,`  
`__gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, type-`  
`name _BinaryFunction2 >`  
`_Tp inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp _`  
`__init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, __`  
`gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename BinaryFunction1, type-`  
`name BinaryFunction2 >`  
`_Tp inner_product (_Iter1, _Iter1, _Iter2, _Tp, BinaryFunction1, Binary-`  
`Function2, __gnu_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, type-`  
`name _BinaryFunction2 >`  
`_Tp inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp _`  
`__init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, __`  
`gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, type-`  
`name _BinaryFunction2 >`  
`_Tp inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init,`  
`_BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __`  
`begin2, _Iter2 __end2, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __`  
`begin2, _Iter2 __end2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __`  
`begin2, _Iter2 __end2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __`  
`begin2, _Iter2 __end2, _Predicate __pred)`
- `template<typename _FIter >`  
`_FIter max_element (_FIter, _FIter, __gnu_parallel::Parallelism)`
- `template<typename _FIter >`  
`_FIter max_element (_FIter, _FIter)`
- `template<typename _FIter >`  
`_FIter max_element (_FIter, _FIter, __gnu_parallel::sequential_tag)`
- `template<typename _FIter, typename _Compare >`  
`_FIter max_element (_FIter, _FIter, _Compare)`
- `template<typename _FIter, typename _Compare >`  
`_FIter max_element (_FIter, _FIter, _Compare, __gnu_parallel::Parallelism)`

- `template<typename _Filter, typename _Compare >`  
`_Filter max_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
  
- `template<typename _FIterator >`  
`_FIterator max_element (_FIterator __begin, _FIterator __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator max_element (_FIterator __begin, _FIterator __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator >`  
`_FIterator max_element (_FIterator __begin, _FIterator __end, \_\_gnu\_parallel::\_Parallelism __parallelism_tag)`
- `template<typename _FIterator >`  
`_FIterator max_element (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator max_element (_FIterator __begin, _FIterator __end, _Compare __comp, \_\_gnu\_parallel::\_Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator max_element (_FIterator __begin, _FIterator __end, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`  
`_OutputIterator merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`  
`_OutputIterator merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Filter >`  
`_Filter min_element (_Filter, _Filter)`

- `template<typename _Filter >`  
`_Filter min_element (_Filter, _Filter, \_\_gnu\_parallel::Parallelism __-`  
`parallelism_tag)`
- `template<typename _Filter, typename _Compare >`  
`_Filter min_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Compare >`  
`_Filter min_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filter >`  
`_Filter min_element (_Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Compare >`  
`_Filter min_element (_Filter, _Filter, _Compare)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator min_element (_FIterator __begin, _FIterator __end, _Compare __-`  
`comp)`
- `template<typename _FIterator >`  
`_FIterator min_element (_FIterator __begin, _FIterator __end, \_\_gnu-`  
`parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator min_element (_FIterator __begin, _FIterator __end, _Compare __-`  
`comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator >`  
`_FIterator min_element (_FIterator __begin, _FIterator __end, \_\_gnu-`  
`parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator >`  
`_FIterator min_element (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator min_element (_FIterator __begin, _FIterator __end, _Compare __-`  
`comp, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`pair< _Iter1, _Iter2 > mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2`  
`__begin2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`pair< _Iter1, _Iter2 > mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2`  
`__begin2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`pair< _Iter1, _Iter2 > mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2`  
`__begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`pair< _Iter1, _Iter2 > mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2`  
`__begin2, _Predicate __pred)`
- `template<typename _RAIter >`  
`void nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, \_\_gnu-`  
`parallel::sequential\_tag)`



- `template<typename _RAIter, typename _Compare >`  
`void nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`  
`void nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end)`
- `template<typename _RAIter, typename _Compare >`  
`void partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`  
`void partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`  
`_OIter partial_sum (_Iter, _Iter, _OIter, _BinaryOper)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter >`  
`_OIter partial_sum (_Iter, _Iter, _OIter __result)`
- `template<typename _Iter, typename _OIter >`  
`_OIter partial_sum (_Iter, _Iter, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`  
`_OIter partial_sum (_Iter, _Iter, _OIter, _BinaryOper, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result)`
- `template<typename _FIter, typename _Predicate >`  
`_FIter partition (_FIter, _FIter, _Predicate)`
- `template<typename _FIter, typename _Predicate >`  
`_FIter partition (_FIter, _FIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`

- template<typename \_FIterator, typename \_Predicate >  
\_FIterator **partition** (\_FIterator \_\_begin, \_FIterator \_\_end, \_Predicate \_\_pred, [\\_\\_gnu\\_parallel::sequential\\_tag](#))
- template<typename \_FIterator, typename \_Predicate >  
\_FIterator **partition** (\_FIterator \_\_begin, \_FIterator \_\_end, \_Predicate \_\_pred)
- template<typename \_RAIter >  
void **random\_shuffle** (\_RAIter \_\_begin, \_RAIter \_\_end, [\\_\\_gnu\\_parallel::sequential\\_tag](#))
- template<typename \_RAIter >  
void **random\_shuffle** (\_RAIter \_\_begin, \_RAIter \_\_end)
- template<typename \_RAIter, typename \_RandomNumberGenerator >  
void **random\_shuffle** (\_RAIter \_\_begin, \_RAIter \_\_end, \_RandomNumberGenerator &&\_\_rand)
- template<typename \_RAIter, typename \_RandomNumberGenerator >  
void **random\_shuffle** (\_RAIter \_\_begin, \_RAIter \_\_end, \_RandomNumberGenerator &\_\_rand, [\\_\\_gnu\\_parallel::sequential\\_tag](#))
- template<typename \_FIterator, typename \_Tp >  
void **replace** (\_FIterator \_\_begin, \_FIterator \_\_end, const \_Tp &\_\_old\_value, const \_Tp &\_\_new\_value)
- template<typename \_FIter, typename \_Tp >  
void **replace** (\_FIter, \_FIter, const \_Tp &, const \_Tp &)
- template<typename \_FIter, typename \_Tp >  
void **replace** (\_FIter, \_FIter, const \_Tp &, const \_Tp &, [\\_\\_gnu\\_parallel::\\_Parallelism](#))
- template<typename \_FIterator, typename \_Tp >  
void **replace** (\_FIterator \_\_begin, \_FIterator \_\_end, const \_Tp &\_\_old\_value, const \_Tp &\_\_new\_value, [\\_\\_gnu\\_parallel::Parallelism](#) \_\_parallelism\_tag)
- template<typename \_FIterator, typename \_Tp >  
void **replace** (\_FIterator \_\_begin, \_FIterator \_\_end, const \_Tp &\_\_old\_value, const \_Tp &\_\_new\_value, [\\_\\_gnu\\_parallel::sequential\\_tag](#))
- template<typename \_FIter, typename \_Tp >  
void **replace** (\_FIter, \_FIter, const \_Tp &, const \_Tp &, [\\_\\_gnu\\_parallel::sequential\\_tag](#))
- template<typename \_FIterator, typename \_Predicate, typename \_Tp >  
void **replace\_if** (\_FIterator \_\_begin, \_FIterator \_\_end, \_Predicate \_\_pred, const \_Tp &\_\_new\_value, [\\_\\_gnu\\_parallel::sequential\\_tag](#))
- template<typename \_FIter, typename \_Predicate, typename \_Tp >  
void **replace\_if** (\_FIter, \_FIter, \_Predicate, const \_Tp &, [\\_\\_gnu\\_parallel::\\_Parallelism](#))
- template<typename \_FIterator, typename \_Predicate, typename \_Tp >  
void **replace\_if** (\_FIterator \_\_begin, \_FIterator \_\_end, \_Predicate \_\_pred, const \_Tp &\_\_new\_value, [\\_\\_gnu\\_parallel::\\_Parallelism](#) \_\_parallelism\_tag)
- template<typename \_FIter, typename \_Predicate, typename \_Tp >  
void **replace\_if** (\_FIter, \_FIter, \_Predicate, const \_Tp &)

- `template<typename _FIterator, typename _Predicate, typename _Tp >`  
`void replace_if (_FIterator __begin, _FIterator __end, _Predicate __pred, const`  
`_Tp &__new_value)`
- `template<typename _FIter, typename _Predicate, typename _Tp >`  
`void replace_if (_FIter, _FIter, _Predicate, const _Tp &, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIter1, typename _FIter2, typename _BiPredicate >`  
`_FIter1 search (_FIter1, _FIter1, _FIter2, _FIter2, _BiPredicate)`
- `template<typename _FIter1, typename _FIter2 >`  
`_FIter1 search (_FIter1, _FIter1, _FIter2, _FIter2)`
- `template<typename _FIter1, typename _FIter2, typename _BiPredicate >`  
`_FIter1 search (_FIter1, _FIter1, _FIter2, _FIter2, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator1, typename _FIterator2 >`  
`_FIterator1 search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __-`  
`begin2, _FIterator2 __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate >`  
`_FIterator1 search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2`  
`__begin2, _FIterator2 __end2, _BinaryPredicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator1, typename _FIterator2 >`  
`_FIterator1 search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __-`  
`begin2, _FIterator2 __end2)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate >`  
`_FIterator1 search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __-`  
`begin2, _FIterator2 __end2, _BinaryPredicate __pred)`
- `template<typename _FIter1, typename _FIter2 >`  
`_FIter1 search (_FIter1, _FIter1, _FIter2, _FIter2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`  
`_FIterator search_n (_FIterator __begin, _FIterator __end, _Integer __-`  
`count, const _Tp &__val, _BinaryPredicate __binary_pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIter, typename _Integer, typename _Tp >`  
`_FIter search_n (_FIter, _FIter, _Integer, const _Tp &)`
- `template<typename _FIter, typename _Integer, typename _Tp, typename _BiPredicate >`  
`_FIter search_n (_FIter, _FIter, _Integer, const _Tp &, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIter, typename _Integer, typename _Tp >`  
`_FIter search_n (_FIter, _FIter, _Integer, const _Tp &, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIter, typename _Integer, typename _Tp, typename _BiPredicate >`  
`_FIter search_n (_FIter, _FIter, _Integer, const _Tp &, _BiPredicate)`

- template<typename \_FIterator, typename \_Integer, typename \_Tp >  
\_FIterator **search\_n** (\_FIterator \_\_begin, \_FIterator \_\_end, \_Integer \_\_count,  
const \_Tp &\_\_val, [\\_\\_gnu\\_parallel::sequential\\_tag](#))
- template<typename \_FIterator, typename \_Integer, typename \_Tp >  
\_FIterator **search\_n** (\_FIterator \_\_begin, \_FIterator \_\_end, \_Integer \_\_count,  
const \_Tp &\_\_val)
- template<typename \_FIterator, typename \_Integer, typename \_Tp, typename \_BinaryPredicate >  
\_FIterator **search\_n** (\_FIterator \_\_begin, \_FIterator \_\_end, \_Integer \_\_count,  
const \_Tp &\_\_val, \_BinaryPredicate \_\_binary\_pred)
- template<typename \_Iter1, typename \_Iter2, typename \_OIter >  
\_OIter **set\_difference** (\_Iter1, \_Iter1, \_Iter2, \_Iter2, \_OIter)
- template<typename \_Iter1, typename \_Iter2, typename \_OIter, typename \_Predicate >  
\_OIter **set\_difference** (\_Iter1, \_Iter1, \_Iter2, \_Iter2, \_OIter, \_Predicate)
- template<typename \_Iter1, typename \_Iter2, typename \_OutputIterator >  
\_OutputIterator **set\_difference** (\_Iter1 \_\_begin1, \_Iter1 \_\_end1, \_Iter2 \_\_-  
begin2, \_Iter2 \_\_end2, \_OutputIterator \_\_out, [\\_\\_gnu\\_parallel::sequential\\_tag](#))
- template<typename \_Iter1, typename \_Iter2, typename \_OutputIterator, typename \_Predicate >  
\_OutputIterator **set\_difference** (\_Iter1 \_\_begin1, \_Iter1 \_\_end1, \_Iter2 \_\_-  
begin2, \_Iter2 \_\_end2, \_OutputIterator \_\_out, \_Predicate \_\_pred, [\\_\\_gnu-  
parallel::sequential\\_tag](#))
- template<typename \_Iter1, typename \_Iter2, typename \_OIter, typename \_Predicate >  
\_OIter **set\_difference** (\_Iter1, \_Iter1, \_Iter2, \_Iter2, \_OIter, \_Predicate, [\\_\\_-  
gnu\\_parallel::sequential\\_tag](#))
- template<typename \_Iter1, typename \_Iter2, typename \_OutputIterator >  
\_OutputIterator **set\_difference** (\_Iter1 \_\_begin1, \_Iter1 \_\_end1, \_Iter2 \_\_-  
begin2, \_Iter2 \_\_end2, \_OutputIterator \_\_out)
- template<typename \_Iter1, typename \_Iter2, typename \_OutputIterator, typename \_Predicate >  
\_OutputIterator **set\_difference** (\_Iter1 \_\_begin1, \_Iter1 \_\_end1, \_Iter2 \_\_-  
begin2, \_Iter2 \_\_end2, \_OutputIterator \_\_out, \_Predicate \_\_pred)
- template<typename \_Iter1, typename \_Iter2, typename \_OIter >  
\_OIter **set\_difference** (\_Iter1, \_Iter1, \_Iter2, \_Iter2, \_OIter, [\\_\\_gnu-  
parallel::sequential\\_tag](#))
- template<typename \_Iter1, typename \_Iter2, typename \_OutputIterator >  
\_OutputIterator **set\_intersection** (\_Iter1 \_\_begin1, \_Iter1 \_\_end1, \_Iter2 \_\_-  
begin2, \_Iter2 \_\_end2, \_OutputIterator \_\_out, [\\_\\_gnu\\_parallel::sequential\\_tag](#))
- template<typename \_Iter1, typename \_Iter2, typename \_OIter >  
\_OIter **set\_intersection** (\_Iter1, \_Iter1, \_Iter2, \_Iter2, \_OIter)
- template<typename \_Iter1, typename \_Iter2, typename \_OIter >  
\_OIter **set\_intersection** (\_Iter1, \_Iter1, \_Iter2, \_Iter2, \_OIter, [\\_\\_gnu-  
parallel::sequential\\_tag](#))
- template<typename \_Iter1, typename \_Iter2, typename \_OutputIterator >  
\_OutputIterator **set\_intersection** (\_Iter1 \_\_begin1, \_Iter1 \_\_end1, \_Iter2 \_\_-  
begin2, \_Iter2 \_\_end2, \_OutputIterator \_\_out)

- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator set_intersection (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`  
`_OIter set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`  
`_OIter set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator set_intersection (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`  
`_OutputIterator set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`  
`_OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`  
`_OutputIterator set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`  
`_OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`  
`_OutputIterator set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,`  
`_Iter2 __end2, _OutputIterator __out)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`  
`_OutputIterator set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,`  
`_Iter2 __end2, _OutputIterator __out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __-`  
`begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred, \_\_gnu-`  
`parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`  
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, \_\_gnu-`  
`parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`  
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _RAIter >`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_-`  
`mergesort\_sampling\_tag __parallelism)`
- `template<typename _RAIter >`  
`void sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter >`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::default\_parallel\_-`  
`tag __parallelism)`
- `template<typename _RAIter >`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_-`  
`mergesort\_exact\_tag __parallelism)`
- `template<typename _RAIter >`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_-`  
`mergesort\_tag __parallelism)`
- `template<typename _RAIter >`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::quicksort\_tag __-`  
`parallelism)`
- `template<typename _RAIter, typename _Compare >`  
`void sort (_RAIter __begin, _RAIter __end, _Compare __comp, \_\_gnu-`  
`parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::balanced\_-`  
`quicksort\_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`  
`void sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::parallel\_tag __-`  
`parallelism)`

- `template<typename _RAIter >`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism >`  
`void sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)`
- `template<typename _RAIter >`  
`void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::balanced\_quicksort\_tag __parallelism)`
- `template<typename _RAIter >`  
`void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_tag __parallelism)`
- `template<typename _RAIter >`  
`void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void stable_sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter >`  
`void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::quicksort\_tag __parallelism)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism >`  
`void stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)`
- `template<typename _RAIter, typename _Compare >`  
`void stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::default\_parallel\_tag __parallelism)`
- `template<typename _RAIter >`  
`void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::parallel\_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`  
`void stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`  
`_OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`  
`_OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`  
`_OIter transform (_Iter, _Iter, _OIter, _UnaryOperation, \_\_gnu\_parallel::\_Parallelism)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation >`  
`_OutputIterator transform (_Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op, \_\_gnu\_parallel::\_Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, \_\_gnu\_parallel::\_Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation >`  
`_OutputIterator transform (_Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`  
`_OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, \_\_gnu\_parallel::\_Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation >`  
`_OutputIterator transform (_Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`  
`_OIter transform (_Iter, _Iter, _OIter, _UnaryOperation, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`  
`_OIter transform (_Iter, _Iter, _OIter, _UnaryOperation)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter >`  
`_OIter unique_copy (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter >`  
`_OIter unique_copy (_Iter, _Iter, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out)`



- `template<typename _Iter, typename _OIter, typename _Predicate >`  
`_OIter unique_copy (_Iter, _Iter, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _Predicate >`  
`_OIter unique_copy (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`

#### 4.14.1 Detailed Description

GNU parallel code, replaces standard behavior with parallel behavior.

### 4.15 `std::__profile` Namespace Reference

GNU profile code, replaces standard behavior with profile behavior.

#### Classes

- class [bitset](#)  
*Class `std::bitset` wrapper with performance instrumentation.*
- class [deque](#)  
*Class `std::deque` wrapper with performance instrumentation.*
- class [forward\\_list](#)  
*Class `std::forward_list` wrapper with performance instrumentation.*
- class [list](#)  
*List wrapper with performance instrumentation.*
- class [map](#)  
*Class `std::map` wrapper with performance instrumentation.*
- class [multimap](#)  
*Class `std::multimap` wrapper with performance instrumentation.*
- class [multiset](#)  
*Class `std::multiset` wrapper with performance instrumentation.*
- class [set](#)  
*Class `std::set` wrapper with performance instrumentation.*

- class [unordered\\_map](#)  
*Class `std::unordered_map` wrapper with performance instrumentation.*
- class [unordered\\_multimap](#)  
*Class `std::unordered_multimap` wrapper with performance instrumentation.*
- class [unordered\\_multiset](#)  
*Unordered\_multiset wrapper with performance instrumentation.*
- class [unordered\\_set](#)  
*Unordered\_set wrapper with performance instrumentation.*

## Functions

- `template<typename _Tp, typename _Alloc >`  
`bool operator!= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator!= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`  
`bool operator!= (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`bool operator!= (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator!= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator!= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`bool operator!= (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator!= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`

- `template<typename _Tp, typename _Alloc >`  
`bool operator!= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator!= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator!= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator!= (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator!= (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`  
`bool operator!= (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<size_t _Nb>`  
`bitset< _Nb > operator& (const bitset< _Nb > &__x, const bitset< _Nb > &__y)`
- `template<typename _Iterator, typename _Sequence >`  
`__iterator_tracker< _Iterator, _Sequence > operator+ (typename __iterator_tracker< _Iterator, _Sequence >::difference_type __n, const __iterator_tracker< _Iterator, _Sequence > &__i)`
- `template<typename _Iterator, typename _Sequence >`  
`__iterator_tracker< _Iterator, _Sequence >::difference_type operator- (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`__iterator_tracker< _IteratorL, _Sequence >::difference_type operator- (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator< (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator< (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator< (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`

- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator< (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator< (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator< (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`  
`std::basic_ostream< _CharT, _Traits > &operator<< (std::basic_ostream< _CharT, _Traits > &__os, const bitset< _Nb > &__x)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator<= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator<= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator<= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator<= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator<= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator<= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator<= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator<= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`

- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator<= (const __iterator_tracker< _IteratorL, _Sequence > &__lhs,`  
`const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator<= (const __iterator_tracker< _Iterator, _Sequence > &__lhs,`  
`const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator== (const set< _Key, _Compare, _Allocator > &__lhs, const set<`  
`_Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator== (const multimap< _Key, _Tp, _Compare, _Allocator > &__-`  
`lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator== (const map< _Key, _Tp, _Compare, _Allocator > &__lhs,`  
`const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`bool operator== (const unordered_multiset< _Value, _Hash, _Pred, _Alloc >`  
`&__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator== (const multiset< _Key, _Compare, _Allocator > &__lhs, const`  
`multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator== (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc >`  
`&__rhs)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`bool operator== (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x,`  
`const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator== (const __iterator_tracker< _Iterator, _Sequence > &__lhs,`  
`const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_-`  
`list< _Tp, _Alloc > &__ly)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator== (const __iterator_tracker< _IteratorL, _Sequence > &__lhs,`  
`const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc`  
`>`  
`bool operator== (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`  
`&__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator== (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, -`  
`_Alloc > &__rhs)`

- `template<typename _Tp, typename _Alloc >`  
`bool operator== (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`  
`bool operator== (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator> (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator> (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator> (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator> (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator> (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator> (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator> (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator> (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator> (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator> (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator>= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator>= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`

- `template<typename _Iterator, typename _Sequence >`  
`bool operator>= (const __iterator_tracker< _Iterator, _Sequence > &__lhs,`  
`const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator>= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp,`  
`_Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator>= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp,`  
`_Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator>= (const forward_list< _Tp, _Alloc > &__lx, const forward_`  
`list< _Tp, _Alloc > &__ly)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator>= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs,`  
`const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator>= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc`  
`> &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator>= (const multimap< _Key, _Tp, _Compare, _Allocator > &__`  
`lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator>= (const __iterator_tracker< _IteratorL, _Sequence > &__lhs,`  
`const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`  
`std::basic_istream< _CharT, _Traits > &operator>> (std::basic_istream< _`  
`CharT, _Traits > &__is, bitset< _Nb > &__x)`
- `template<size_t _Nb>`  
`bitset< _Nb > operator^ (const bitset< _Nb > &__x, const bitset< _Nb >`  
`&__y)`
- `template<size_t _Nb>`  
`bitset< _Nb > operator| (const bitset< _Nb > &__x, const bitset< _Nb >`  
`&__y)`
- `template<typename _Tp, typename _Alloc >`  
`void swap (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &&__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`void swap (deque< _Tp, _Alloc > &__lhs, deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`void swap (multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, mul-`  
`timap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`void swap (vector< _Tp, _Alloc > &&__lhs, vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`void swap (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &__rhs)`

- `template<typename _Tp, typename _Alloc >`  
`void swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc >`  
`&__ly)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`void swap (set< _Key, _Compare, _Allocator > &__x, set< _Key, _Compare,`  
`_Allocator > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`void swap (list< _Tp, _Alloc > &__lhs, list< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc`  
`>`  
`void swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x,`  
`unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc`  
`>`  
`void swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x,`  
`unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`void swap (map< _Key, _Tp, _Compare, _Allocator > &__lhs, map< _Key,`  
`_Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`void swap (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_-`  
`set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`void swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x,`  
`unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`void swap (multiset< _Key, _Compare, _Allocator > &__x, multiset< _Key,`  
`_Compare, _Allocator > &__y)`

#### 4.15.1 Detailed Description

GNU profile code, replaces standard behavior with profile behavior.

#### 4.15.2 Function Documentation

- ##### 4.15.2.1 `template<typename _Tp, typename _Alloc > bool` `std::__profile::operator<= ( const forward_list< _Tp, _Alloc > &` `__lx, const forward_list< _Tp, _Alloc > & __ly ) [inline]`

Based on operator<.

Definition at line 163 of file profile/forward\_list.



**4.15.2.2** `template<typename _Tp , typename _Alloc > bool  
std::__profile::operator> ( const forward_list< _Tp, _Alloc > & __lx,  
const forward_list< _Tp, _Alloc > & __ly ) [inline]`

Based on `operator<`.

Definition at line 149 of file `profile/forward_list`.

**4.15.2.3** `template<typename _Tp , typename _Alloc > bool  
std::__profile::operator>= ( const forward_list< _Tp, _Alloc > &  
__lx, const forward_list< _Tp, _Alloc > & __ly ) [inline]`

Based on `operator<`.

Definition at line 156 of file `profile/forward_list`.

**4.15.2.4** `template<typename _Tp , typename _Alloc > void std::__profile::swap  
( forward_list< _Tp, _Alloc > & __lx, forward_list< _Tp, _Alloc >  
& __ly ) [inline]`

See `std::forward_list::swap()`.

Definition at line 170 of file `profile/forward_list`.

## 4.16 `std::chrono` Namespace Reference

ISO C++ 0x entities sub namespace for time and date.

### Classes

- struct `duration`  
*duration*
- struct `duration_values`  
*duration\_values*
- struct `system_clock`  
*system\_clock*

- struct [time\\_point](#)  
*time\_point*
- struct [treat\\_as\\_floating\\_point](#)  
*treat\_as\_floating\_point*

### Typedefs

- typedef [system\\_clock](#) **high\_resolution\_clock**
- typedef [duration](#)< int, [ratio](#)< 3600 > > **hours**
- typedef [duration](#)< int64\_t, micro > **microseconds**
- typedef [duration](#)< int64\_t, milli > **milliseconds**
- typedef [duration](#)< int, [ratio](#)< 60 > > **minutes**
- typedef [system\\_clock](#) **monotonic\_clock**
- typedef [duration](#)< int64\_t, nano > **nanoseconds**
- typedef [duration](#)< int64\_t > **seconds**

### Functions

- template<typename \_ToDur, typename \_Rep, typename \_Period >  
constexpr [enable\\_if](#)< \_\_is\_duration< \_ToDur >::value, \_ToDur >::type  
[duration\\_cast](#) (const [duration](#)< \_Rep, \_Period > &\_\_d)
- template<typename \_Clock, typename \_Dur1, typename \_Dur2 >  
constexpr bool **operator!=** (const [time\\_point](#)< \_Clock, \_Dur1 > &\_\_lhs, const  
[time\\_point](#)< \_Clock, \_Dur2 > &\_\_rhs)
- template<typename \_Rep1, typename \_Period1, typename \_Rep2, typename \_Period2 >  
constexpr bool **operator!=** (const [duration](#)< \_Rep1, \_Period1 > &\_\_lhs, const  
[duration](#)< \_Rep2, \_Period2 > &\_\_rhs)
- template<typename \_Rep1, typename \_Period1, typename \_Rep2, typename \_Period2 >  
common\_type< [duration](#)< \_Rep1, \_Period1 >, [duration](#)< \_Rep2, \_Period2 >  
>::type **operator%** (const [duration](#)< \_Rep1, \_Period1 > &\_\_lhs, const [duration](#)< \_Rep2, \_Period2 > &\_\_rhs)
- template<typename \_Rep1, typename \_Period, typename \_Rep2 >  
[duration](#)< typename \_\_common\_rep\_type< \_Rep1, typename [enable\\_if](#)<!\_\_is\_duration< \_Rep2 >::value, \_Rep2 >::type >::type, \_Period > **operator%**  
(const [duration](#)< \_Rep1, \_Period > &\_\_d, const \_Rep2 &\_\_s)
- template<typename \_Rep1, typename \_Period, typename \_Rep2 >  
[duration](#)< typename \_\_common\_rep\_type< \_Rep1, \_Rep2 >::type, \_Period >  
**operator\*** (const [duration](#)< \_Rep1, \_Period > &\_\_d, const \_Rep2 &\_\_s)
- template<typename \_Rep1, typename \_Period, typename \_Rep2 >  
[duration](#)< typename \_\_common\_rep\_type< \_Rep2, \_Rep1 >::type, \_Period >  
**operator\*** (const \_Rep1 &\_\_s, const [duration](#)< \_Rep2, \_Period > &\_\_d)

- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >  
common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 >  
>::type operator+ (const duration< _Rep1, _Period1 > &__lhs, const dura-  
tion< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2 >  
time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _-  
Period2 > >::type > operator+ (const time_point< _Clock, _Dur1 > &__lhs,  
const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Clock, typename _Dur2 >  
time_point< _Clock, typename common_type< duration< _Rep1, _Period1 >,  
_Dur2 >::type > operator+ (const duration< _Rep1, _Period1 > &__lhs, const  
time_point< _Clock, _Dur2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2 >  
time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _-  
Period2 > >::type > operator- (const time_point< _Clock, _Dur1 > &__lhs,  
const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2 >  
common_type< _Dur1, _Dur2 >::type operator- (const time_point< _Clock,  
_Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >  
common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 >  
>::type operator- (const duration< _Rep1, _Period1 > &__lhs, const dura-  
tion< _Rep2, _Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period, typename _Rep2 >  
duration< typename __common_rep_type< _Rep1, typename enable_if<!__-  
is_duration< _Rep2 >::value, _Rep2 >::type >::type, _Period > operator/  
(const duration< _Rep1, _Period > &__d, const _Rep2 &__s)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >  
common_type< _Rep1, _Rep2 >::type operator/ (const duration< _Rep1, _-  
Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2 >  
constexpr bool operator< (const time_point< _Clock, _Dur1 > &__lhs, const  
time_point< _Clock, _Dur2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >  
constexpr bool operator< (const duration< _Rep1, _Period1 > &__lhs, const  
duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >  
constexpr bool operator<= (const duration< _Rep1, _Period1 > &__lhs, const  
duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2 >  
constexpr bool operator<= (const time_point< _Clock, _Dur1 > &__lhs, const  
time_point< _Clock, _Dur2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2 >  
constexpr bool operator== (const time_point< _Clock, _Dur1 > &__lhs, const  
time_point< _Clock, _Dur2 > &__rhs)`

- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`constexpr bool operator== (const duration< _Rep1, _Period1 > &__lhs, const`  
`duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2 >`  
`constexpr bool operator> (const time_point< _Clock, _Dur1 > &__lhs, const`  
`time_point< _Clock, _Dur2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`constexpr bool operator> (const duration< _Rep1, _Period1 > &__lhs, const`  
`duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2 >`  
`constexpr bool operator>= (const time_point< _Clock, _Dur1 > &__lhs, const`  
`time_point< _Clock, _Dur2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`constexpr bool operator>= (const duration< _Rep1, _Period1 > &__lhs, const`  
`duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _ToDur, typename _Clock, typename _Dur >`  
`constexpr enable_if< __is_duration< _ToDur >::value, time_point< _Clock,`  
`_ToDur >::type time_point_cast (const time_point< _Clock, _Dur > &__t)`

#### 4.16.1 Detailed Description

ISO C++ 0x entities sub namespace for time and date.

#### 4.16.2 Typedef Documentation

##### 4.16.2.1 typedef duration<int, ratio<3600> > std::chrono::hours

hours

Definition at line 514 of file chrono.

##### 4.16.2.2 typedef duration<int64\_t, micro> std::chrono::microseconds

microseconds

Definition at line 502 of file chrono.

##### 4.16.2.3 typedef duration<int64\_t, milli> std::chrono::milliseconds

milliseconds

Definition at line 505 of file chrono.

#### 4.16.2.4 typedef duration<int, ratio< 60> > std::chrono::minutes

minutes

Definition at line 511 of file chrono.

#### 4.16.2.5 typedef duration<int64\_t, nano> std::chrono::nanoseconds

nanoseconds

Definition at line 499 of file chrono.

#### 4.16.2.6 typedef duration<int64\_t> std::chrono::seconds

seconds

Definition at line 508 of file chrono.

### 4.16.3 Function Documentation

#### 4.16.3.1 template<typename \_ToDur , typename \_Rep , typename \_Period > constexpr enable\_if<\_\_is\_duration<\_ToDur>::value, \_ToDur>::type std::chrono::duration\_cast ( const duration< \_Rep, \_Period > & \_\_d ) [inline]

duration\_cast

Definition at line 173 of file chrono.

Referenced by std::this\_thread::sleep\_for().

**4.16.3.2** `template<typename _ToDur , typename _Clock , typename  
_Dur > constexpr enable_if<__is_duration<_ToDur>::value,  
time_point<_Clock, _ToDur> >::type std::chrono::time_point_cast (  
const time_point< _Clock, _Dur > & __t ) [inline]`

`time_point_cast`

Definition at line 575 of file `chrono`.

## 4.17 `std::decimal` Namespace Reference

ISO/IEC TR 24733 Decimal floating-point arithmetic.

### Classes

- class [decimal128](#)  
3.2.4 Class *decimal128*.
- class [decimal32](#)  
3.2.2 Class *decimal32*.
- class [decimal64](#)  
3.2.3 Class *decimal64*.

### Functions

- double **decimal128\_to\_double** ([decimal128](#) \_\_d)
- float **decimal128\_to\_float** ([decimal128](#) \_\_d)
- long double **decimal128\_to\_long\_double** ([decimal128](#) \_\_d)
- long long **decimal128\_to\_long\_long** ([decimal128](#) \_\_d)
- double **decimal32\_to\_double** ([decimal32](#) \_\_d)
- float **decimal32\_to\_float** ([decimal32](#) \_\_d)
- long double **decimal32\_to\_long\_double** ([decimal32](#) \_\_d)
- long long **decimal32\_to\_long\_long** ([decimal32](#) \_\_d)
- double **decimal64\_to\_double** ([decimal64](#) \_\_d)
- float **decimal64\_to\_float** ([decimal64](#) \_\_d)
- long double **decimal64\_to\_long\_double** ([decimal64](#) \_\_d)
- long long **decimal64\_to\_long\_long** ([decimal64](#) \_\_d)
- double **decimal\_to\_double** ([decimal32](#) \_\_d)
- double **decimal\_to\_double** ([decimal64](#) \_\_d)

- double **decimal\_to\_double** ([decimal128](#) \_\_d)
- float **decimal\_to\_float** ([decimal32](#) \_\_d)
- float **decimal\_to\_float** ([decimal64](#) \_\_d)
- float **decimal\_to\_float** ([decimal128](#) \_\_d)
- long double **decimal\_to\_long\_double** ([decimal32](#) \_\_d)
- long double **decimal\_to\_long\_double** ([decimal64](#) \_\_d)
- long double **decimal\_to\_long\_double** ([decimal128](#) \_\_d)
- long long **decimal\_to\_long\_long** ([decimal32](#) \_\_d)
- long long **decimal\_to\_long\_long** ([decimal64](#) \_\_d)
- long long **decimal\_to\_long\_long** ([decimal128](#) \_\_d)
- static [decimal128](#) **make\_decimal128** (long long \_\_coeff, int \_\_exp)
- static [decimal128](#) **make\_decimal128** (unsigned long long \_\_coeff, int \_\_exp)
- static [decimal32](#) **make\_decimal32** (long long \_\_coeff, int \_\_exp)
- static [decimal32](#) **make\_decimal32** (unsigned long long \_\_coeff, int \_\_exp)
- static [decimal64](#) **make\_decimal64** (unsigned long long \_\_coeff, int \_\_exp)
- static [decimal64](#) **make\_decimal64** (long long \_\_coeff, int \_\_exp)
- bool **operator!=** ([decimal32](#) \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, int \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, unsigned int \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, long \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, unsigned long \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, long long \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, unsigned long long \_\_rhs)
- bool **operator!=** (int \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator!=** (unsigned int \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator!=** (long \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator!=** (unsigned long \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator!=** (long long \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator!=** (unsigned long long \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator!=** ([decimal64](#) \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator!=** ([decimal64](#) \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator!=** ([decimal64](#) \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator!=** ([decimal64](#) \_\_lhs, int \_\_rhs)
- bool **operator!=** ([decimal64](#) \_\_lhs, unsigned int \_\_rhs)
- bool **operator!=** ([decimal64](#) \_\_lhs, long \_\_rhs)
- bool **operator!=** ([decimal64](#) \_\_lhs, unsigned long \_\_rhs)
- bool **operator!=** ([decimal64](#) \_\_lhs, long long \_\_rhs)
- bool **operator!=** ([decimal64](#) \_\_lhs, unsigned long long \_\_rhs)
- bool **operator!=** (int \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator!=** (unsigned int \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator!=** (long \_\_lhs, [decimal64](#) \_\_rhs)

- bool **operator!=** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **operator!=** (long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator!=** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator!=** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- bool **operator!=** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- bool **operator!=** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **operator!=** (decimal128 \_\_lhs, int \_\_rhs)
- bool **operator!=** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **operator!=** (decimal128 \_\_lhs, long \_\_rhs)
- bool **operator!=** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- bool **operator!=** (decimal128 \_\_lhs, long long \_\_rhs)
- bool **operator!=** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator!=** (int \_\_lhs, decimal128 \_\_rhs)
- bool **operator!=** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **operator!=** (long \_\_lhs, decimal128 \_\_rhs)
- bool **operator!=** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **operator!=** (long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator!=** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- decimal32 **operator\*** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- decimal32 **operator\*** (decimal32 \_\_lhs, int \_\_rhs)
- decimal32 **operator\*** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- decimal32 **operator\*** (decimal32 \_\_lhs, long \_\_rhs)
- decimal32 **operator\*** (decimal32 \_\_lhs, long long \_\_rhs)
- decimal32 **operator\*** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- decimal32 **operator\*** (int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **operator\*** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **operator\*** (long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **operator\*** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **operator\*** (long long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **operator\*** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- decimal64 **operator\*** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **operator\*** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- decimal64 **operator\*** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **operator\*** (decimal64 \_\_lhs, int \_\_rhs)
- decimal64 **operator\*** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- decimal64 **operator\*** (decimal64 \_\_lhs, long \_\_rhs)
- decimal64 **operator\*** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- decimal64 **operator\*** (decimal64 \_\_lhs, long long \_\_rhs)
- decimal64 **operator\*** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- decimal64 **operator\*** (int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **operator\*** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **operator\*** (long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **operator\*** (unsigned long \_\_lhs, decimal64 \_\_rhs)



- [decimal64 operator\\*](#) (long long \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator\\*](#) (unsigned long long \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal128 operator\\*](#) ([decimal32](#) \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator\\*](#) ([decimal64](#) \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator\\*](#) ([decimal128](#) \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal128 operator\\*](#) ([decimal128](#) \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal128 operator\\*](#) ([decimal128](#) \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator\\*](#) ([decimal128](#) \_\_lhs, int \_\_rhs)
- [decimal128 operator\\*](#) ([decimal128](#) \_\_lhs, unsigned int \_\_rhs)
- [decimal128 operator\\*](#) ([decimal128](#) \_\_lhs, long \_\_rhs)
- [decimal128 operator\\*](#) ([decimal128](#) \_\_lhs, unsigned long \_\_rhs)
- [decimal128 operator\\*](#) ([decimal128](#) \_\_lhs, long long \_\_rhs)
- [decimal128 operator\\*](#) ([decimal128](#) \_\_lhs, unsigned long long \_\_rhs)
- [decimal128 operator\\*](#) (int \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator\\*](#) (unsigned int \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator\\*](#) (long \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator\\*](#) (unsigned long \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator\\*](#) (long long \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator\\*](#) (unsigned long long \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal32 operator\\*](#) ([decimal32](#) \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal64 operator+](#) (unsigned long long \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator+](#) ([decimal64](#) \_\_rhs)
- [decimal128 operator+](#) ([decimal32](#) \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator+](#) ([decimal64](#) \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator+](#) ([decimal128](#) \_\_rhs)
- [decimal128 operator+](#) ([decimal128](#) \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal128 operator+](#) ([decimal128](#) \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal128 operator+](#) ([decimal128](#) \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator+](#) ([decimal128](#) \_\_lhs, int \_\_rhs)
- [decimal128 operator+](#) ([decimal128](#) \_\_lhs, unsigned int \_\_rhs)
- [decimal128 operator+](#) ([decimal128](#) \_\_lhs, long \_\_rhs)
- [decimal128 operator+](#) ([decimal128](#) \_\_lhs, unsigned long \_\_rhs)
- [decimal128 operator+](#) ([decimal128](#) \_\_lhs, long long \_\_rhs)
- [decimal32 operator+](#) ([decimal32](#) \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal128 operator+](#) ([decimal128](#) \_\_lhs, unsigned long long \_\_rhs)
- [decimal128 operator+](#) (int \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal32 operator+](#) ([decimal32](#) \_\_lhs, int \_\_rhs)
- [decimal128 operator+](#) (unsigned int \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator+](#) (long \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal32 operator+](#) ([decimal32](#) \_\_lhs, unsigned int \_\_rhs)
- [decimal128 operator+](#) (unsigned long \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator+](#) (long long \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal32 operator+](#) ([decimal32](#) \_\_lhs, long \_\_rhs)

- [decimal128 operator+](#) (unsigned long long \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal32 operator+](#) ([decimal32](#) \_\_lhs, unsigned long \_\_rhs)
- [decimal32 operator+](#) ([decimal32](#) \_\_lhs, long long \_\_rhs)
- [decimal32 operator+](#) ([decimal32](#) \_\_lhs, unsigned long long \_\_rhs)
- [decimal32 operator+](#) (int \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal32 operator+](#) (unsigned int \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal32 operator+](#) (long \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal32 operator+](#) (unsigned long \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal32 operator+](#) (long long \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal32 operator+](#) (unsigned long long \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal64 operator+](#) ([decimal32](#) \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator+](#) ([decimal64](#) \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal64 operator+](#) ([decimal64](#) \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator+](#) ([decimal64](#) \_\_lhs, int \_\_rhs)
- [decimal64 operator+](#) ([decimal64](#) \_\_lhs, unsigned int \_\_rhs)
- [decimal64 operator+](#) ([decimal64](#) \_\_lhs, long \_\_rhs)
- [decimal64 operator+](#) ([decimal64](#) \_\_lhs, unsigned long \_\_rhs)
- [decimal64 operator+](#) ([decimal64](#) \_\_lhs, long long \_\_rhs)
- [decimal64 operator+](#) ([decimal64](#) \_\_lhs, unsigned long long \_\_rhs)
- [decimal64 operator+](#) (int \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator+](#) (unsigned int \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator+](#) (long \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator+](#) (unsigned long \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal32 operator+](#) ([decimal32](#) \_\_rhs)
- [decimal64 operator+](#) (long long \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal32 operator-](#) ([decimal32](#) \_\_rhs)
- [decimal64 operator-](#) ([decimal64](#) \_\_rhs)
- [decimal128 operator-](#) ([decimal128](#) \_\_rhs)
- [decimal32 operator-](#) ([decimal32](#) \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal32 operator-](#) ([decimal32](#) \_\_lhs, int \_\_rhs)
- [decimal32 operator-](#) ([decimal32](#) \_\_lhs, unsigned int \_\_rhs)
- [decimal32 operator-](#) ([decimal32](#) \_\_lhs, long \_\_rhs)
- [decimal32 operator-](#) ([decimal32](#) \_\_lhs, unsigned long \_\_rhs)
- [decimal32 operator-](#) ([decimal32](#) \_\_lhs, long long \_\_rhs)
- [decimal32 operator-](#) ([decimal32](#) \_\_lhs, unsigned long long \_\_rhs)
- [decimal32 operator-](#) (int \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal32 operator-](#) (unsigned int \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal32 operator-](#) (long \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal32 operator-](#) (unsigned long \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal32 operator-](#) (long long \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal32 operator-](#) (unsigned long long \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal64 operator-](#) ([decimal32](#) \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator-](#) ([decimal64](#) \_\_lhs, [decimal32](#) \_\_rhs)

- **decimal64 operator-** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- **decimal64 operator-** (decimal64 \_\_lhs, int \_\_rhs)
- **decimal64 operator-** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- **decimal64 operator-** (decimal64 \_\_lhs, long \_\_rhs)
- **decimal64 operator-** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- **decimal64 operator-** (decimal64 \_\_lhs, long long \_\_rhs)
- **decimal64 operator-** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- **decimal64 operator-** (int \_\_lhs, decimal64 \_\_rhs)
- **decimal64 operator-** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- **decimal64 operator-** (long \_\_lhs, decimal64 \_\_rhs)
- **decimal64 operator-** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- **decimal64 operator-** (long long \_\_lhs, decimal64 \_\_rhs)
- **decimal64 operator-** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- **decimal128 operator-** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator-** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator-** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- **decimal128 operator-** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- **decimal128 operator-** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator-** (decimal128 \_\_lhs, int \_\_rhs)
- **decimal128 operator-** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- **decimal128 operator-** (decimal128 \_\_lhs, long \_\_rhs)
- **decimal128 operator-** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- **decimal128 operator-** (decimal128 \_\_lhs, long long \_\_rhs)
- **decimal128 operator-** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- **decimal128 operator-** (int \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator-** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator-** (long \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator-** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator-** (long long \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator-** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- **decimal32 operator/** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- **decimal32 operator/** (decimal32 \_\_lhs, int \_\_rhs)
- **decimal32 operator/** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- **decimal32 operator/** (decimal32 \_\_lhs, long \_\_rhs)
- **decimal32 operator/** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- **decimal32 operator/** (decimal32 \_\_lhs, long long \_\_rhs)
- **decimal32 operator/** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- **decimal32 operator/** (int \_\_lhs, decimal32 \_\_rhs)
- **decimal32 operator/** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- **decimal32 operator/** (long \_\_lhs, decimal32 \_\_rhs)
- **decimal32 operator/** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- **decimal128 operator/** (long long \_\_lhs, decimal128 \_\_rhs)
- **decimal32 operator/** (long long \_\_lhs, decimal32 \_\_rhs)

- **decimal32 operator/** (unsigned long long \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal64 operator/** ([decimal32](#) \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal64 operator/** ([decimal64](#) \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal64 operator/** ([decimal64](#) \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal64 operator/** ([decimal64](#) \_\_lhs, int \_\_rhs)
- **decimal64 operator/** ([decimal64](#) \_\_lhs, unsigned int \_\_rhs)
- **decimal128 operator/** ([decimal128](#) \_\_lhs, long \_\_rhs)
- **decimal64 operator/** ([decimal64](#) \_\_lhs, long \_\_rhs)
- **decimal64 operator/** ([decimal64](#) \_\_lhs, unsigned long \_\_rhs)
- **decimal64 operator/** ([decimal64](#) \_\_lhs, long long \_\_rhs)
- **decimal128 operator/** ([decimal128](#) \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal64 operator/** ([decimal64](#) \_\_lhs, unsigned long long \_\_rhs)
- **decimal64 operator/** (int \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal64 operator/** (unsigned int \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal64 operator/** (long \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal64 operator/** (unsigned long \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal64 operator/** (long long \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal64 operator/** (unsigned long long \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal128 operator/** ([decimal32](#) \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal128 operator/** ([decimal64](#) \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal128 operator/** ([decimal128](#) \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal128 operator/** ([decimal128](#) \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal128 operator/** ([decimal128](#) \_\_lhs, int \_\_rhs)
- **decimal128 operator/** ([decimal128](#) \_\_lhs, unsigned int \_\_rhs)
- **decimal128 operator/** ([decimal128](#) \_\_lhs, unsigned long \_\_rhs)
- **decimal128 operator/** ([decimal128](#) \_\_lhs, long long \_\_rhs)
- **decimal128 operator/** ([decimal128](#) \_\_lhs, unsigned long long \_\_rhs)
- **decimal128 operator/** (int \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal128 operator/** (unsigned int \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal128 operator/** (long \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal128 operator/** (unsigned long \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal128 operator/** (unsigned long long \_\_lhs, [decimal128](#) \_\_rhs)
- **bool operator<** ([decimal128](#) \_\_lhs, [decimal32](#) \_\_rhs)
- **bool operator<** ([decimal64](#) \_\_lhs, [decimal32](#) \_\_rhs)
- **bool operator<** (int \_\_lhs, [decimal32](#) \_\_rhs)
- **bool operator<** (unsigned long \_\_lhs, [decimal32](#) \_\_rhs)
- **bool operator<** ([decimal32](#) \_\_lhs, unsigned long long \_\_rhs)
- **bool operator<** (unsigned int \_\_lhs, [decimal32](#) \_\_rhs)
- **bool operator<** (long \_\_lhs, [decimal32](#) \_\_rhs)
- **bool operator<** ([decimal32](#) \_\_lhs, unsigned int \_\_rhs)
- **bool operator<** (unsigned int \_\_lhs, [decimal128](#) \_\_rhs)
- **bool operator<** ([decimal32](#) \_\_lhs, long \_\_rhs)
- **bool operator<** ([decimal64](#) \_\_lhs, unsigned long long \_\_rhs)

- bool **operator**< (decimal32 \_\_lhs, long long \_\_rhs)
- bool **operator**< (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**< (long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**< (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**< (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**< (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**< (long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**< (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**< (decimal64 \_\_lhs, int \_\_rhs)
- bool **operator**< (decimal32 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**< (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**< (decimal128 \_\_lhs, long \_\_rhs)
- bool **operator**< (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**< (decimal128 \_\_lhs, int \_\_rhs)
- bool **operator**< (decimal128 \_\_lhs, long long \_\_rhs)
- bool **operator**< (decimal32 \_\_lhs, int \_\_rhs)
- bool **operator**< (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**< (int \_\_lhs, decimal128 \_\_rhs)
- bool **operator**< (decimal32 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**< (decimal128 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**< (decimal32 \_\_lhs, decimal32 \_\_rhs)
- bool **operator**< (int \_\_lhs, decimal64 \_\_rhs)
- bool **operator**< (long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**< (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**< (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**< (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**< (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **operator**< (long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**< (long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**< (decimal64 \_\_lhs, long \_\_rhs)
- bool **operator**< (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**< (decimal128 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**< (decimal64 \_\_lhs, long long \_\_rhs)
- bool **operator**< (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**== (int \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, decimal128 \_\_rhs)

- bool **operator==** (decimal128 \_\_lhs, long long \_\_rhs)
- bool **operator==** (decimal128 \_\_lhs, long \_\_rhs)
- bool **operator==** (decimal32 \_\_lhs, long \_\_rhs)
- bool **operator==** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **operator==** (decimal64 \_\_lhs, int \_\_rhs)
- bool **operator==** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **operator==** (long \_\_lhs, decimal64 \_\_rhs)
- bool **operator==** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- bool **operator==** (decimal64 \_\_lhs, long long \_\_rhs)
- bool **operator==** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **operator==** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **operator==** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **operator==** (long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator==** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **operator==** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **operator==** (long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator==** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- bool **operator==** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- bool **operator==** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator==** (decimal32 \_\_lhs, int \_\_rhs)
- bool **operator==** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- bool **operator==** (decimal32 \_\_lhs, long long \_\_rhs)
- bool **operator==** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator==** (int \_\_lhs, decimal32 \_\_rhs)
- bool **operator==** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **operator==** (long \_\_lhs, decimal32 \_\_rhs)
- bool **operator==** (decimal64 \_\_lhs, long \_\_rhs)
- bool **operator==** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **operator==** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **operator==** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator==** (int \_\_lhs, decimal64 \_\_rhs)
- bool **operator==** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **operator==** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **operator==** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- bool **operator==** (decimal128 \_\_lhs, int \_\_rhs)
- bool **operator==** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator>** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- bool **operator>** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **operator>** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **operator>** (long \_\_lhs, decimal32 \_\_rhs)
- bool **operator>** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **operator>** (decimal128 \_\_lhs, int \_\_rhs)
- bool **operator>** (decimal32 \_\_lhs, unsigned long \_\_rhs)

- bool **operator**> (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**> (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **operator**> (int \_\_lhs, decimal64 \_\_rhs)
- bool **operator**> (decimal32 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**> (decimal32 \_\_lhs, long long \_\_rhs)
- bool **operator**> (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**> (decimal32 \_\_lhs, int \_\_rhs)
- bool **operator**> (decimal32 \_\_lhs, decimal32 \_\_rhs)
- bool **operator**> (long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**> (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**> (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**> (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**> (long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**> (int \_\_lhs, decimal128 \_\_rhs)
- bool **operator**> (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**> (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**> (long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**> (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**> (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **operator**> (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**> (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**> (long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**> (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**> (decimal128 \_\_lhs, long \_\_rhs)
- bool **operator**> (decimal64 \_\_lhs, long \_\_rhs)
- bool **operator**> (decimal128 \_\_lhs, decimal32 \_\_rhs)
- bool **operator**> (decimal128 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**> (decimal64 \_\_lhs, int \_\_rhs)
- bool **operator**> (decimal128 \_\_lhs, long long \_\_rhs)
- bool **operator**> (decimal128 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**> (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**> (decimal64 \_\_lhs, long long \_\_rhs)
- bool **operator**> (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **operator**> (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**> (long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**> (decimal32 \_\_lhs, long \_\_rhs)
- bool **operator**> (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**> (int \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, int \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, int \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, long \_\_rhs)

- bool **operator**>= (decimal32 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, long \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, long long \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**>= (long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, int \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, long long \_\_rhs)
- bool **operator**>= (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**>= (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (int \_\_lhs, decimal128 \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**>= (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**>= (long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**>= (int \_\_lhs, decimal64 \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, long long \_\_rhs)
- bool **operator**>= (int \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**>= (long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**>= (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**>= (long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**>= (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **operator**>= (long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, long \_\_rhs)
- bool **operator**>= (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, decimal64 \_\_rhs)



### 4.17.1 Detailed Description

ISO/IEC TR 24733 Decimal floating-point arithmetic.

### 4.17.2 Function Documentation

#### 4.17.2.1 `long long std::decimal::decimal32_to_long_long ( decimal32 __d )`

Non-conforming extension: Conversion to integral type.

## 4.18 `std::placeholders` Namespace Reference

ISO C++ 0x entities sub namespace for functional.

Define a large number of placeholders. There is no way to simplify this with variadic templates, because we're introducing unique names for each.

### Variables

- `const _Placeholder< 1 > _1`
- `const _Placeholder< 10 > _10`
- `const _Placeholder< 11 > _11`
- `const _Placeholder< 12 > _12`
- `const _Placeholder< 13 > _13`
- `const _Placeholder< 14 > _14`
- `const _Placeholder< 15 > _15`
- `const _Placeholder< 16 > _16`
- `const _Placeholder< 17 > _17`
- `const _Placeholder< 18 > _18`
- `const _Placeholder< 19 > _19`
- `const _Placeholder< 2 > _2`
- `const _Placeholder< 20 > _20`
- `const _Placeholder< 21 > _21`
- `const _Placeholder< 22 > _22`
- `const _Placeholder< 23 > _23`
- `const _Placeholder< 24 > _24`
- `const _Placeholder< 25 > _25`
- `const _Placeholder< 26 > _26`
- `const _Placeholder< 27 > _27`
- `const _Placeholder< 28 > _28`
- `const _Placeholder< 29 > _29`

- `const _Placeholder< 3 > _3`
- `const _Placeholder< 4 > _4`
- `const _Placeholder< 5 > _5`
- `const _Placeholder< 6 > _6`
- `const _Placeholder< 7 > _7`
- `const _Placeholder< 8 > _8`
- `const _Placeholder< 9 > _9`

#### 4.18.1 Detailed Description

ISO C++ 0x entities sub namespace for functional.

Define a large number of placeholders. There is no way to simplify this with variadic templates, because we're introducing unique names for each.

### 4.19 `std::regex_constants` Namespace Reference

ISO C++-0x entities sub namespace for regex.

#### 5.1 Regular Expression Syntax Options

- `enum __syntax_option {`  
    `_S_icode, _S_nosubs, _S_optimize, _S_collate,`  
    `_S_ECMAScript, _S_basic, _S_extended, _S_awk,`  
    `_S_grep, _S_egrep, _S_syntax_last }`
- `typedef unsigned int syntax_option_type`
- `static const syntax_option_type icode`
- `static const syntax_option_type nosubs`
- `static const syntax_option_type optimize`
- `static const syntax_option_type collate`
- `static const syntax_option_type ECMAScript`
- `static const syntax_option_type basic`
- `static const syntax_option_type extended`
- `static const syntax_option_type awk`
- `static const syntax_option_type grep`
- `static const syntax_option_type egrep`

## 5.2 Matching Rules

Matching a regular expression against a sequence of characters [first, last) proceeds according to the rules of the grammar specified for the regular expression object, modified according to the effects listed below for any bitmask elements set.

- enum `__match_flag` {  
`_S_not_bol`, `_S_not_eol`, `_S_not_bow`, `_S_not_eow`,  
`_S_any`, `_S_not_null`, `_S_continuous`, `_S_prev_avail`,  
`_S_sed`, `_S_no_copy`, `_S_first_only`, `_S_match_flag_last` }
- typedef `std::bitset<_S_match_flag_last>` `match_flag_type`
- static const `match_flag_type` `match_default`
- static const `match_flag_type` `match_not_bol`
- static const `match_flag_type` `match_not_eol`
- static const `match_flag_type` `match_not_bow`
- static const `match_flag_type` `match_not_eow`
- static const `match_flag_type` `match_any`
- static const `match_flag_type` `match_not_null`
- static const `match_flag_type` `match_continuous`
- static const `match_flag_type` `match_prev_avail`
- static const `match_flag_type` `format_default`
- static const `match_flag_type` `format_sed`
- static const `match_flag_type` `format_no_copy`
- static const `match_flag_type` `format_first_only`

## 5.3 Error Types

- enum `error_type` {  
`_S_error_collate`, `_S_error_ctype`, `_S_error_escape`, `_S_error_backref`,  
`_S_error_brack`, `_S_error_paren`, `_S_error_brace`, `_S_error_badbrace`,  
`_S_error_range`, `_S_error_space`, `_S_error_badrepeat`, `_S_error_-complexity`,  
`_S_error_stack`, `_S_error_last` }
- static const `error_type` `error_collate` (`_S_error_collate`)
- static const `error_type` `error_ctype` (`_S_error_ctype`)
- static const `error_type` `error_escape` (`_S_error_escape`)
- static const `error_type` `error_backref` (`_S_error_backref`)
- static const `error_type` `error_brack` (`_S_error_brack`)
- static const `error_type` `error_paren` (`_S_error_paren`)
- static const `error_type` `error_brace` (`_S_error_brace`)
- static const `error_type` `error_badbrace` (`_S_error_badbrace`)

- static const `error_type error_range` (`_S_error_range`)
- static const `error_type error_space` (`_S_error_space`)
- static const `error_type error_badrepeat` (`_S_error_badrepeat`)
- static const `error_type error_complexity` (`_S_error_complexity`)
- static const `error_type error_stack` (`_S_error_stack`)

#### 4.19.1 Detailed Description

ISO C++-0x entities sub namespace for regex.

#### 4.19.2 Typedef Documentation

##### 4.19.2.1 `typedef std::bitset<_S_match_flag_last> std::regex_constants::match_flag_type`

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

Definition at line 191 of file `regex_constants.h`.

##### 4.19.2.2 `typedef unsigned int std::regex_constants::syntax_option_type`

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

Definition at line 73 of file `regex_constants.h`.

#### 4.19.3 Enumeration Type Documentation

##### 4.19.3.1 `enum std::regex_constants::__match_flag`

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

Definition at line 168 of file `regex_constants.h`.

#### 4.19.3.2 `enum std::regex_constants::__syntax_option`

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

Definition at line 47 of file `regex_constants.h`.

#### 4.19.3.3 `enum std::regex_constants::error_type`

The expression contained an invalid collating element name.

Definition at line 44 of file `regex_error.h`.

### 4.19.4 Function Documentation

#### 4.19.4.1 `static const error_type std::regex_constants::error_backref ( _S_error_backref ) [static]`

The expression contained an invalid back reference.

#### 4.19.4.2 `static const error_type std::regex_constants::error_badbrace ( _S_error_badbrace ) [static]`

The expression contained an invalid range in a `{ }` expression.

#### 4.19.4.3 `static const error_type std::regex_constants::error_badrepeat ( _S_error_badrepeat ) [static]`

One of `*?+{` was not preceded by a valid regular expression.

**4.19.4.4** `static const error_type std::regex_constants::error_brace (`  
`_S_error_brace ) [static]`

The expression contained mismatched { and }

**4.19.4.5** `static const error_type std::regex_constants::error_brack (`  
`_S_error_brack ) [static]`

The expression contained mismatched [ and ].

**4.19.4.6** `static const error_type std::regex_constants::error_collate (`  
`_S_error_collate ) [static]`

The expression contained an invalid collating element name.

**4.19.4.7** `static const error_type std::regex_constants::error_complexity (`  
`_S_error_complexity ) [static]`

The complexity of an attempted match against a regular expression exceeded a pre-set level.

**4.19.4.8** `static const error_type std::regex_constants::error_ctype (`  
`_S_error_ctype ) [static]`

The expression contained an invalid character class name.

**4.19.4.9** `static const error_type std::regex_constants::error_escape (`  
`_S_error_escape ) [static]`

The expression contained an invalid escaped character, or a trailing escape.

**4.19.4.10** `static const error_type std::regex_constants::error_paren (`  
`_S_error_paren ) [static]`

The expression contained mismatched ( and ).

**4.19.4.11** `static const error_type std::regex_constants::error_range (`  
`_S_error_range ) [static]`

The expression contained an invalid character range, such as [b-a] in most encodings.

#### 4.19.4.12 `static const error_type std::regex_constants::error_space (` `_S_error_space ) [static]`

There was insufficient memory to convert the expression into a finite state machine.

#### 4.19.4.13 `static const error_type std::regex_constants::error_stack (` `_S_error_stack ) [static]`

There was insufficient memory to determine whether the regular expression could match the specified character sequence.

### 4.19.5 Variable Documentation

#### 4.19.5.1 `const syntax_option_type std::regex_constants::awk [static]`

Specifies that the grammar recognized by the regular expression engine is that used by POSIX utility `awk` in IEEE Std 1003.1-2001. This option is identical to `syntax_option_type extended`, except that C-style escape sequences are supported. These sequences are: `\\`, `\a`, `\b`, `\f`, `\n`, `\r`, `\t`, `\v`, `\'`, `\`, and `\ddd` (where `ddd` is one, two, or three octal digits).

Definition at line 137 of file `regex_constants.h`.

#### 4.19.5.2 `const syntax_option_type std::regex_constants::basic [static]`

Specifies that the grammar recognized by the regular expression engine is that used by POSIX basic regular expressions in IEEE Std 1003.1-2001, Portable Operating System Interface (POSIX), Base Definitions and Headers, Section 9, Regular Expressions [IEEE, Information Technology -- Portable Operating System Interface (POSIX), IEEE Standard 1003.1-2001].

Definition at line 119 of file `regex_constants.h`.

#### 4.19.5.3 `const syntax_option_type std::regex_constants::collate [static]`

Specifies that character ranges of the form `[a-b]` should be locale sensitive.

Definition at line 100 of file `regex_constants.h`.

#### 4.19.5.4 `const syntax_option_type std::regex_constants::ECMAScript` `[static]`

Specifies that the grammar recognized by the regular expression engine is that used by ECMAScript in ECMA-262 [Ecma International, ECMAScript Language

Specification, Standard Ecma-262, third edition, 1999], as modified in section [28.13]. This grammar is similar to that defined in the PERL scripting language but extended with elements found in the POSIX regular expression grammar.

Definition at line 110 of file `regex_constants.h`.

#### 4.19.5.5 `const syntax_option_type std::regex_constants::egrep` `[static]`

Specifies that the grammar recognized by the regular expression engine is that used by POSIX utility `grep` when given the `-E` option in IEEE Std 1003.1-2001. This option is identical to `syntax_option_type` extended, except that newlines are treated as whitespace.

Definition at line 153 of file `regex_constants.h`.

#### 4.19.5.6 `const syntax_option_type std::regex_constants::extended` `[static]`

Specifies that the grammar recognized by the regular expression engine is that used by POSIX extended regular expressions in IEEE Std 1003.1-2001, Portable Operating System Interface (POSIX), Base Definitions and Headers, Section 9, Regular Expressions.

Definition at line 127 of file `regex_constants.h`.

#### 4.19.5.7 `const match_flag_type std::regex_constants::format_default` `[static]`

When a regular expression match is to be replaced by a new string, the new string is constructed using the rules used by the ECMAScript `replace` function in ECMA-262 [Ecma International, ECMAScript Language Specification, Standard Ecma-262, third edition, 1999], part 15.5.4.11 `String.prototype.replace`. In addition, during search and replace operations all non-overlapping occurrences of the regular expression are located and replaced, and sections of the input that did not match the expression are copied unchanged to the output string.

Format strings (from ECMA-262 [15.5.4.11]):

- `$$` The dollar-sign itself (`$`)
- `$&` The matched substring.
- `$'` The portion of *string* that precedes the matched substring. This would be [`match\_results::prefix\(\)`](#).
- `$'` The portion of *string* that follows the matched substring. This would be [`match\_results::suffix\(\)`](#).



- `$n` The *n*th capture, where *n* is in [1,9] and `$n` is not followed by a decimal digit. If *n*  $\leq$  `match_results::size()` and the *n*th capture is undefined, use the empty string instead. If *n*  $>$  `match_results::size()`, the result is implementation-defined.
- `$nn` The *nn*th capture, where *nn* is a two-digit decimal number on [01, 99]. If *nn*  $\leq$  `match_results::size()` and the *n*th capture is undefined, use the empty string instead. If *nn*  $>$  `match_results::size()`, the result is implementation-defined.

Definition at line 273 of file `regex_constants.h`.

#### 4.19.5.8 `const match_flag_type std::regex_constants::format_first_only` `[static]`

When specified during a search and replace operation, only the first occurrence of the regular expression shall be replaced.

Definition at line 294 of file `regex_constants.h`.

#### 4.19.5.9 `const match_flag_type std::regex_constants::format_no_copy` `[static]`

During a search and replace operation, sections of the character container sequence being searched that do not match the regular expression shall not be copied to the output string.

Definition at line 288 of file `regex_constants.h`.

#### 4.19.5.10 `const match_flag_type std::regex_constants::format_sed` `[static]`

When a regular expression match is to be replaced by a new string, the new string is constructed using the rules used by the POSIX `sed` utility in IEEE Std 1003.1- 2001 [IEEE, Information Technology -- Portable Operating System Interface (POSIX), IEEE Standard 1003.1-2001].

Definition at line 281 of file `regex_constants.h`.

#### 4.19.5.11 `const syntax_option_type std::regex_constants::grep` `[static]`

Specifies that the grammar recognized by the regular expression engine is that used by POSIX utility `grep` in IEEE Std 1003.1-2001. This option is identical to `syntax_option_type basic`, except that newlines are treated as whitespace.

Definition at line 145 of file `regex_constants.h`.

**4.19.5.12 `const syntax_option_type std::regex_constants::icase [static]`**

Specifies that the matching of regular expressions against a character sequence shall be performed without regard to case.

Definition at line 79 of file `regex_constants.h`.

**4.19.5.13 `const match_flag_type std::regex_constants::match_any [static]`**

If more than one match is possible then any match is an acceptable result.

Definition at line 228 of file `regex_constants.h`.

**4.19.5.14 `const match_flag_type std::regex_constants::match_continuous [static]`**

The expression only matches a sub-sequence that begins at first .

Definition at line 238 of file `regex_constants.h`.

**4.19.5.15 `const match_flag_type std::regex_constants::match_default [static]`**

The default matching rules.

Definition at line 196 of file `regex_constants.h`.

**4.19.5.16 `const match_flag_type std::regex_constants::match_not_bol [static]`**

The first character in the sequence [first, last) is treated as though it is not at the beginning of a line, so the character (^) in the regular expression shall not match [first, first).

Definition at line 203 of file `regex_constants.h`.

**4.19.5.17 `const match_flag_type std::regex_constants::match_not_bow [static]`**

The expression \b is not matched against the sub-sequence [first,first).

Definition at line 216 of file `regex_constants.h`.

**4.19.5.18 `const match_flag_type std::regex_constants::match_not_eol`  
`[static]`**

The last character in the sequence `[first, last)` is treated as though it is not at the end of a line, so the character `(\n)` in the regular expression shall not match `[last, last)`.

Definition at line 210 of file `regex_constants.h`.

**4.19.5.19 `const match_flag_type std::regex_constants::match_not_eow`  
`[static]`**

The expression `\b` should not be matched against the sub-sequence `[last,last)`.

Definition at line 222 of file `regex_constants.h`.

**4.19.5.20 `const match_flag_type std::regex_constants::match_not_null`  
`[static]`**

The expression does not match an empty sequence.

Definition at line 233 of file `regex_constants.h`.

**4.19.5.21 `const match_flag_type std::regex_constants::match_prev_avail`  
`[static]`**

`--first` is a valid iterator position. When this flag is set then the flags `match_not_bol` and `match_not_bow` are ignored by the regular expression algorithms 7.11 and iterators 7.12.

Definition at line 245 of file `regex_constants.h`.

**4.19.5.22 `const syntax_option_type std::regex_constants::nosubs` `[static]`**

Specifies that when a regular expression is matched against a character container sequence, no sub-expression matches are to be stored in the supplied `match_results` structure.

Definition at line 86 of file `regex_constants.h`.

**4.19.5.23 `const syntax_option_type std::regex_constants::optimize` `[static]`**

Specifies that the regular expression engine should pay more attention to the speed with which regular expressions are matched, and less to the speed with which regular expression objects are constructed. Otherwise it has no detectable effect on the program output.

Definition at line 94 of file `regex_constants.h`.

## 4.20 `std::rel_ops` Namespace Reference

The generated relational operators are sequestered here.

### Functions

- `template<class _Tp >`  
`bool operator!= (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp >`  
`bool operator<= (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp >`  
`bool operator> (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp >`  
`bool operator>= (const _Tp &__x, const _Tp &__y)`

### 4.20.1 Detailed Description

The generated relational operators are sequestered here.

### 4.20.2 Function Documentation

#### 4.20.2.1 `template<class _Tp > bool std::rel_ops::operator!= ( const _Tp & __x, const _Tp & __y ) [inline]`

Defines `!=` for arbitrary types, in terms of `==`.

#### Parameters

- x* A thing.
- y* Another thing.

#### Returns

`x != y`

This function uses `==` to determine its result.

Definition at line 88 of file `stl_relops.h`.

**4.20.2.2** `template<class _Tp > bool std::rel_ops::operator<= ( const _Tp & __x, const _Tp & __y ) [inline]`

Defines `<=` for arbitrary types, in terms of `<`.

#### Parameters

- x* A thing.
- y* Another thing.

#### Returns

`x <= y`

This function uses `<` to determine its result.

Definition at line 114 of file `stl_relops.h`.

**4.20.2.3** `template<class _Tp > bool std::rel_ops::operator> ( const _Tp & __x, const _Tp & __y ) [inline]`

Defines `>` for arbitrary types, in terms of `<`.

#### Parameters

- x* A thing.
- y* Another thing.

#### Returns

`x > y`

This function uses `<` to determine its result.

Definition at line 101 of file `stl_relops.h`.

**4.20.2.4** `template<class _Tp > bool std::rel_ops::operator>= ( const _Tp & __x, const _Tp & __y ) [inline]`

Defines `>=` for arbitrary types, in terms of `<`.

**Parameters**

- x* A thing.
- y* Another thing.

**Returns**

`x >= y`

This function uses `<` to determine its result.

Definition at line 127 of file `std_relops.h`.

**4.21 `std::this_thread` Namespace Reference**

ISO C++ 0x entities sub namespace for thread. 30.2.2 Namespace [this\\_thread](#).

**Functions**

- [thread::id](#) `get_id()`
- `template<typename _Rep, typename _Period>`  
`void` [sleep\\_for](#) (`const` [chrono::duration](#)< `_Rep`, `_Period` > &`__rtime`)
- `template<typename _Clock, typename _Duration>`  
`void` [sleep\\_until](#) (`const` [chrono::time\\_point](#)< `_Clock`, `_Duration` > &`__atime`)
- `void` [yield](#) ()

**4.21.1 Detailed Description**

ISO C++ 0x entities sub namespace for thread. 30.2.2 Namespace [this\\_thread](#).

**4.21.2 Function Documentation****4.21.2.1 `thread::id` `std::this_thread::get_id()` [`inline`]**

`get_id`

Definition at line 253 of file `thread`.

**4.21.2.2 `template<typename _Rep, typename _Period> void`  
`std::this_thread::sleep_for` ( `const` `chrono::duration`< `_Rep`, `_Period`  
 > & `__rtime` ) [`inline`]**

sleep\_for

Definition at line 272 of file thread.

References std::chrono::duration\_cast().

Referenced by sleep\_until().

**4.21.2.3** `template<typename _Clock , typename _Duration > void  
std::this_thread::sleep_until ( const chrono::time_point< _Clock,  
_Duration > & __atime ) [inline]`

sleep\_until

Definition at line 266 of file thread.

References sleep\_for().

**4.21.2.4** `void std::this_thread::yield ( ) [inline]`

yield

Definition at line 258 of file thread.

## 4.22 std::tr1 Namespace Reference

ISO C++ TR1 entities toplevel namespace is [std::tr1](#).

### Namespaces

- namespace [\\_\\_detail](#)

### Functions

- `template<typename _Tp >  
std::complex< _Tp > __complex_acos (const std::complex< _Tp > &__z)`
- `template<typename _Tp >  
std::complex< _Tp > __complex_acosh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >  
std::complex< _Tp > __complex_asin (const std::complex< _Tp > &__z)`
- `template<typename _Tp >  
std::complex< _Tp > __complex_asinh (const std::complex< _Tp > &__z)`

- template<typename \_Tp >  
std::complex< \_Tp > \_\_complex\_atan (const std::complex< \_Tp > &\_\_z)
- template<typename \_Tp >  
std::complex< \_Tp > \_\_complex\_atanh (const std::complex< \_Tp > &\_\_z)
- template<typename \_Tp >  
std::complex< \_Tp > acos (const std::complex< \_Tp > &\_\_z)
- template<typename \_Tp >  
std::complex< \_Tp > acosh (const std::complex< \_Tp > &\_\_z)
- template<typename \_Tp >  
std::complex< \_Tp > asin (const std::complex< \_Tp > &\_\_z)
- template<typename \_Tp >  
std::complex< \_Tp > asinh (const std::complex< \_Tp > &\_\_z)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type assoc\_laguerre (unsigned int \_\_n, unsigned int \_\_m, \_Tp \_\_x)
- float assoc\_laguerref (unsigned int \_\_n, unsigned int \_\_m, float \_\_x)
- long double assoc\_laguerrel (unsigned int \_\_n, unsigned int \_\_m, long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type assoc\_legendre (unsigned int \_\_l, unsigned int \_\_m, \_Tp \_\_x)
- float assoc\_legendref (unsigned int \_\_l, unsigned int \_\_m, float \_\_x)
- long double assoc\_legendrel (unsigned int \_\_l, unsigned int \_\_m, long double \_\_x)
- template<typename \_Tp >  
std::complex< \_Tp > atan (const std::complex< \_Tp > &\_\_z)
- template<typename \_Tp >  
std::complex< \_Tp > atanh (const std::complex< \_Tp > &\_\_z)
- template<typename \_Tpx, typename \_Tpy >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tpx, \_Tpy >::\_\_type beta (\_Tpx \_\_x, \_Tpy \_\_y)
- float betaf (float \_\_x, float \_\_y)
- long double betal (long double \_\_x, long double \_\_y)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type comp\_ellint\_1 (\_Tp \_\_k)
- float comp\_ellint\_1f (float \_\_k)
- long double comp\_ellint\_1l (long double \_\_k)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type comp\_ellint\_2 (\_Tp \_\_k)
- float comp\_ellint\_2f (float \_\_k)
- long double comp\_ellint\_2l (long double \_\_k)
- template<typename \_Tp, typename \_Tpn >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Tpn >::\_\_type comp\_ellint\_3 (\_Tp \_\_k, \_Tpn \_\_nu)
- float comp\_ellint\_3f (float \_\_k, float \_\_nu)



- long double **comp\_ellint\_3l** (long double \_\_k, long double \_\_nu)
- template<typename \_Tpa, typename \_Tpc, typename \_Tp >  
\_\_gnu\_cxx::\_\_promote\_3< \_Tpa, \_Tpc, \_Tp >::\_\_type **conf\_hyperg** (\_Tpa \_\_a,  
\_Tpc \_\_c, \_Tp \_\_x)
- float **conf\_hypergf** (float \_\_a, float \_\_c, float \_\_x)
- long double **conf\_hypergl** (long double \_\_a, long double \_\_c, long double \_\_x)
- template<typename \_Tp >  
[std::complex](#)< typename \_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type > **conj** (\_Tp  
\_\_x)
- template<typename \_Tpnu, typename \_Tp >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tpnu, \_Tp >::\_\_type **cyl\_bessel\_i** (\_Tpnu \_\_nu,  
\_Tp \_\_x)
- float **cyl\_bessel\_if** (float \_\_nu, float \_\_x)
- long double **cyl\_bessel\_il** (long double \_\_nu, long double \_\_x)
- template<typename \_Tpnu, typename \_Tp >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tpnu, \_Tp >::\_\_type **cyl\_bessel\_j** (\_Tpnu \_\_nu,  
\_Tp \_\_x)
- float **cyl\_bessel\_jf** (float \_\_nu, float \_\_x)
- long double **cyl\_bessel\_jl** (long double \_\_nu, long double \_\_x)
- template<typename \_Tpnu, typename \_Tp >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tpnu, \_Tp >::\_\_type **cyl\_bessel\_k** (\_Tpnu \_\_nu,  
\_Tp \_\_x)
- float **cyl\_bessel\_kf** (float \_\_nu, float \_\_x)
- long double **cyl\_bessel\_kl** (long double \_\_nu, long double \_\_x)
- template<typename \_Tpnu, typename \_Tp >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tpnu, \_Tp >::\_\_type **cyl\_neumann** (\_Tpnu \_\_nu,  
\_Tp \_\_x)
- float **cyl\_neumannf** (float \_\_nu, float \_\_x)
- long double **cyl\_neumannl** (long double \_\_nu, long double \_\_x)
- template<typename \_Tp, typename \_Tpp >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Tpp >::\_\_type **ellint\_1** (\_Tp \_\_k, \_Tpp \_\_-  
phi)
- float **ellint\_1f** (float \_\_k, float \_\_phi)
- long double **ellint\_1l** (long double \_\_k, long double \_\_phi)
- template<typename \_Tp, typename \_Tpp >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Tpp >::\_\_type **ellint\_2** (\_Tp \_\_k, \_Tpp \_\_-  
phi)
- float **ellint\_2f** (float \_\_k, float \_\_phi)
- long double **ellint\_2l** (long double \_\_k, long double \_\_phi)
- template<typename \_Tp, typename \_Tpn, typename \_Tpp >  
\_\_gnu\_cxx::\_\_promote\_3< \_Tp, \_Tpn, \_Tpp >::\_\_type **ellint\_3** (\_Tp \_\_k, \_Tpn  
\_\_nu, \_Tpp \_\_phi)
- float **ellint\_3f** (float \_\_k, float \_\_nu, float \_\_phi)
- long double **ellint\_3l** (long double \_\_k, long double \_\_nu, long double \_\_phi)

- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **expint** (\_Tp \_\_x)
- float **expintf** (float \_\_x)
- long double **expintl** (long double \_\_x)
- template<typename \_Tp >  
[std::complex](#)< \_Tp > **fabs** (const [std::complex](#)< \_Tp > &\_\_z)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **hermite** (unsigned int \_\_n, \_Tp \_\_x)
- float **hermitef** (unsigned int \_\_n, float \_\_x)
- long double **hermitel** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tpa, typename \_Tpb, typename \_Tpc, typename \_Tp >  
\_\_gnu\_cxx::\_\_promote\_4< \_Tpa, \_Tpb, \_Tpc, \_Tp >::\_\_type **hyperg** (\_Tpa \_\_a, \_Tpb \_\_b, \_Tpc \_\_c, \_Tp \_\_x)
- float **hypergfl** (float \_\_a, float \_\_b, float \_\_c, float \_\_x)
- long double **hypergl** (long double \_\_a, long double \_\_b, long double \_\_c, long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **laguerre** (unsigned int \_\_n, \_Tp \_\_x)
- float **laguerref** (unsigned int \_\_n, float \_\_x)
- long double **laguerrel** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **legendre** (unsigned int \_\_n, \_Tp \_\_x)
- float **legendref** (unsigned int \_\_n, float \_\_x)
- long double **legendrel** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp, typename \_Up >  
[std::complex](#)< typename \_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Up >::\_\_type > **polar** (const \_Tp &\_\_rho, const \_Up &\_\_theta)
- template<typename \_Tp, typename \_Up >  
[std::complex](#)< typename \_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Up >::\_\_type > **pow** (const [std::complex](#)< \_Tp > &\_\_x, const [std::complex](#)< \_Up > &\_\_y)
- long double **pow** (long double \_\_x, long double \_\_y)
- double **pow** (double \_\_x, double \_\_y)
- template<typename \_Tp, typename \_Up >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Up >::\_\_type **pow** (\_Tp \_\_x, \_Up \_\_y)
- float **pow** (float \_\_x, float \_\_y)
- template<typename \_Tp, typename \_Up >  
[std::complex](#)< typename \_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Up >::\_\_type > **pow** (const \_Tp &\_\_x, const [std::complex](#)< \_Up > &\_\_y)
- template<typename \_Tp, typename \_Up >  
[std::complex](#)< typename \_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Up >::\_\_type > **pow** (const [std::complex](#)< \_Tp > &\_\_x, const \_Up &\_\_y)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **riemann\_zeta** (\_Tp \_\_x)

- float **riemann\_zetaf** (float \_\_x)
- long double **riemann\_zetal** (long double \_\_x)
- template<typename \_Tp >  
  \_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **sph\_bessel** (unsigned int \_\_n, \_Tp \_\_x)
- float **sph\_besself** (unsigned int \_\_n, float \_\_x)
- long double **sph\_bessell** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp >  
  \_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **sph\_legendre** (unsigned int \_\_l, unsigned int \_\_m, \_Tp \_\_theta)
- float **sph\_legendref** (unsigned int \_\_l, unsigned int \_\_m, float \_\_theta)
- long double **sph\_legendrel** (unsigned int \_\_l, unsigned int \_\_m, long double \_\_theta)
- template<typename \_Tp >  
  \_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **sph\_neumann** (unsigned int \_\_n, \_Tp \_\_x)
- float **sph\_neumannf** (unsigned int \_\_n, float \_\_x)
- long double **sph\_neumannl** (unsigned int \_\_n, long double \_\_x)

#### 4.22.1 Detailed Description

ISO C++ TR1 entities toplevel namespace is [std::tr1](#).

### 4.23 `std::tr1::__detail` Namespace Reference

Implementation details not part of the namespace [std::tr1](#) interface.

#### 4.23.1 Detailed Description

Implementation details not part of the namespace [std::tr1](#) interface.

## 5 Class Documentation

### 5.1 `__cxxabiv1::__forced_unwind` Class Reference

Thrown as part of forced unwinding.

A magic placeholder class that can be caught by reference to recognize forced unwinding.

## 5.2 `__gnu_cxx::__common_pool_policy<_PoolTp, _Thread>` Struct Template Reference 817

---

### 5.1.1 Detailed Description

Thrown as part of forced unwinding.

A magic placeholder class that can be caught by reference to recognize forced unwinding.

Definition at line 48 of file `cxxabi_forced.h`.

The documentation for this class was generated from the following file:

- [cxxabi\\_forced.h](#)

## 5.2 `__gnu_cxx::__common_pool_policy<_PoolTp, _Thread>` Struct Template Reference

Policy for shared `__pool` objects.

Inherits `__common_pool_base<_PoolTp, _Thread>`.

### 5.2.1 Detailed Description

`template<template< bool > class _PoolTp, bool _Thread> struct __gnu_cxx::__common_pool_policy<_PoolTp, _Thread>`

Policy for shared `__pool` objects.

Definition at line 458 of file `mt_allocator.h`.

The documentation for this struct was generated from the following file:

- [mt\\_allocator.h](#)

## 5.3 `__gnu_cxx::__detail::__mini_vector<_Tp>` Class Template Reference

`__mini_vector<>` is a stripped down version of the full-fledged `std::vector<>`.

### Public Types

- `typedef const _Tp & const_reference`
- `typedef ptrdiff_t difference_type`
- `typedef pointer iterator`
- `typedef _Tp * pointer`
- `typedef _Tp & reference`

## 5.4 `__gnu_cxx::__detail::_Bitmap_counter<_Tp>` Class Template Reference 818

- typedef `size_t` **size\_type**
- typedef `_Tp` **value\_type**

### Public Member Functions

- reference **back** () const throw ()
- [iterator](#) **begin** () const throw ()
- void **clear** () throw ()
- [iterator](#) **end** () const throw ()
- void **erase** ([iterator](#) \_\_pos) throw ()
- void **insert** ([iterator](#) \_\_pos, const\_reference \_\_x)
- reference **operator**[ ] (const size\_type \_\_pos) const throw ()
- void **pop\_back** () throw ()
- void **push\_back** (const\_reference \_\_x)
- size\_type **size** () const throw ()

### 5.3.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::__detail::_mini_vector<_Tp>`

`_mini_vector<>` is a stripped down version of the full-fledged `std::vector<>`. It is to be used only for built-in types or PODs. Notable differences are:

1. Not all accessor functions are present. 2. Used ONLY for PODs. 3. No Allocator template argument. Uses [operator new\(\)](#) to get memory, and [operator delete\(\)](#) to free it. Caveat: The dtor does NOT free the memory allocated, so this a memory-leaking vector!

Definition at line 71 of file `bitmap_allocator.h`.

The documentation for this class was generated from the following file:

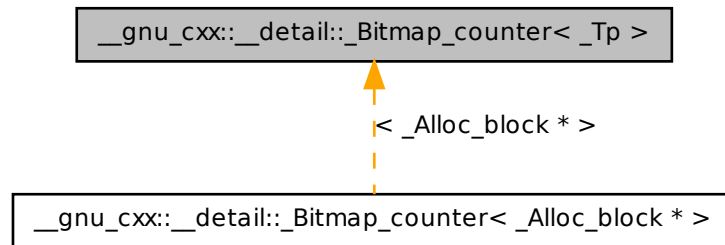
- [bitmap\\_allocator.h](#)

## 5.4 `__gnu_cxx::__detail::_Bitmap_counter<_Tp>` Class Template Reference

The bitmap counter which acts as the bitmap manipulator, and manages the bit-manipulation functions and the searching and identification functions on the bit-map.

## 5.4 `__gnu_cxx::__detail::_Bitmap_counter<_Tp>` Class Template Reference

Inheritance diagram for `__gnu_cxx::__detail::_Bitmap_counter<_Tp>`:



### Public Member Functions

- `_Bitmap_counter` (`_BPVector` &Rvbp, long `__index=-1`)
- `pointer _M_base` () const throw ()
- `bool _M_finished` () const throw ()
- `size_t * _M_get` () const throw ()
- `_Index_type _M_offset` () const throw ()
- `void _M_reset` (long `__index=-1`) throw ()
- `void _M_set_internal_bitmap` (size\_t \*`__new_internal_marker`) throw ()
- `_Index_type _M_where` () const throw ()
- `_Bitmap_counter` & `operator++` () throw ()

#### 5.4.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::__detail::_Bitmap_counter<_Tp>`

The bitmap counter which acts as the bitmap manipulator, and manages the bit-manipulation functions and the searching and identification functions on the bit-map.

Definition at line 400 of file `bitmap_allocator.h`.

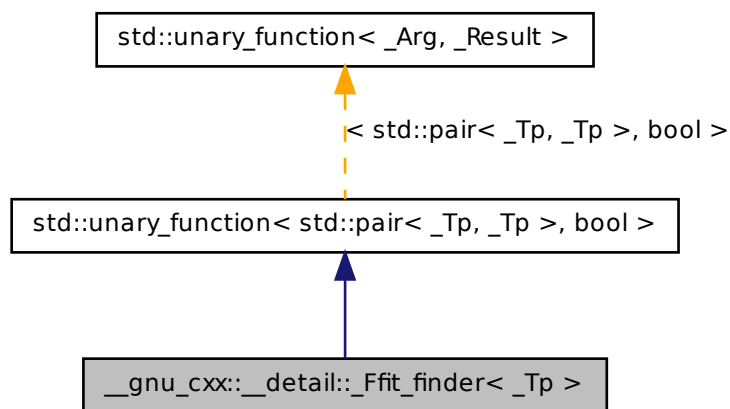
The documentation for this class was generated from the following file:

- [bitmap\\_allocator.h](#)

## 5.5 `__gnu_cxx::__detail::_Ffit_finder<_Tp>` Class Template Reference

The class which acts as a predicate for applying the first-fit memory allocation policy for the bitmap allocator.

Inheritance diagram for `__gnu_cxx::__detail::_Ffit_finder<_Tp>`:



### Public Types

- typedef `std::pair<_Tp, _Tp>` `argument_type`
- typedef `bool` `result_type`

### Public Member Functions

- `size_t * _M_get () const throw ()`
- `_Counter_type _M_offset () const throw ()`
- `bool operator() (_Block_pair __bp) throw ()`

### 5.5.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::__detail::_Ffit_finder<_Tp>`

The class which acts as a predicate for applying the first-fit memory allocation policy for the bitmap allocator.

Definition at line 335 of file `bitmap_allocator.h`.

### 5.5.2 Member Typedef Documentation

**5.5.2.1** `typedef std::pair<_Tp, _Tp> std::unary_function< std::pair<_Tp, _Tp>, bool>::argument_type [inherited]`

`argument_type` is the type of the argument

Definition at line 105 of file `stl_function.h`.

**5.5.2.2** `typedef bool std::unary_function< std::pair<_Tp, _Tp>, bool>::result_type [inherited]`

`result_type` is the return type

Definition at line 108 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [bitmap\\_allocator.h](#)

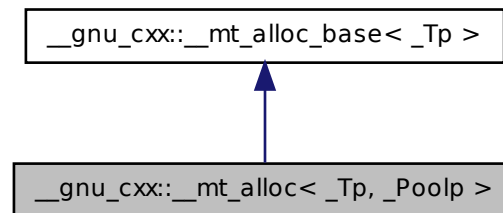
## 5.6 `__gnu_cxx::__mt_alloc<_Tp, _Poolp>` Class Template Reference

This is a fixed size (power of 2) allocator which - when compiled with thread support - will maintain one freelist per size per thread plus a *global* one. Steps are taken to limit the per thread freelist sizes (by returning excess back to the *global* list).

Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt12ch32.html>.



Inheritance diagram for `__gnu_cxx::__mt_alloc<_Tp, _Poolp>`:



### Public Types

- `typedef _Poolp __policy_type`
- `typedef _Poolp::pool_type __pool_type`
- `typedef const _Tp * const_pointer`
- `typedef const _Tp & const_reference`
- `typedef ptrdiff_t difference_type`
- `typedef _Tp * pointer`
- `typedef _Tp & reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

### Public Member Functions

- `__mt_alloc` (`const __mt_alloc &`) `throw ()`
- `template<typename _Tp1, typename _Poolp1 >`  
`__mt_alloc` (`const __mt_alloc<_Tp1, _Poolp1> &`) `throw ()`
- `const __pool_base::_Tune _M_get_options ()`
- `void _M_set_options (__pool_base::_Tune __t)`
- `pointer address` (`reference __x`) `const`
- `const_pointer address` (`const_reference __x`) `const`
- `pointer allocate` (`size_type __n, const void * = 0`)
- `template<typename... _Args>`  
`void construct` (`pointer __p, _Args &&... __args`)
- `void construct` (`pointer __p, const _Tp & __val`)
- `void deallocate` (`pointer __p, size_type __n`)

- void **destroy** (pointer \_\_p)
- size\_type **max\_size** () const throw ()

### 5.6.1 Detailed Description

template<typename \_Tp, typename \_Poolp = \_\_common\_pool\_policy<\_\_pool, true >> class \_\_gnu\_cxx::\_\_mt\_alloc< \_Tp, \_Poolp >

This is a fixed size (power of 2) allocator which - when compiled with thread support - will maintain one freelist per size per thread plus a *global* one. Steps are taken to limit the per thread freelist sizes (by returning excess back to the *global* list).

Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt12ch32.html>.

Definition at line 627 of file mt\_allocator.h.

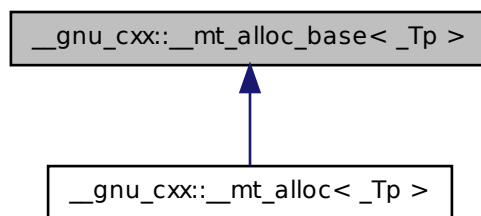
The documentation for this class was generated from the following file:

- [mt\\_allocator.h](#)

## 5.7 \_\_gnu\_cxx::\_\_mt\_alloc\_base< \_Tp > Class Template Reference

Base class for \_Tp dependent member functions.

Inheritance diagram for \_\_gnu\_cxx::\_\_mt\_alloc\_base< \_Tp >:



### Public Types

- typedef const \_Tp \* **const\_pointer**

- typedef const \_Tp & **const\_reference**
- typedef ptrdiff\_t **difference\_type**
- typedef \_Tp \* **pointer**
- typedef \_Tp & **reference**
- typedef size\_t **size\_type**
- typedef \_Tp **value\_type**

### Public Member Functions

- pointer **address** (reference \_\_x) const
- const\_pointer **address** (const\_reference \_\_x) const
- template<typename... \_Args>  
void **construct** (pointer \_\_p, \_Args &&...\_\_args)
- void **construct** (pointer \_\_p, const \_Tp &\_\_val)
- void **destroy** (pointer \_\_p)
- size\_type **max\_size** () const throw ()

#### 5.7.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::__mt_alloc_base< _Tp >`

Base class for \_Tp dependent member functions.

Definition at line 568 of file `mt_allocator.h`.

The documentation for this class was generated from the following file:

- [mt\\_allocator.h](#)

### 5.8 `__gnu_cxx::__per_type_pool_policy< _Tp, _PoolTp, _Thread > Struct` **Template Reference**

Policy for individual \_\_pool objects.

Inherits `__per_type_pool_base< _Tp, _PoolTp, _Thread >`.

#### 5.8.1 Detailed Description

`template<typename _Tp, template< bool > class _PoolTp, bool _Thread> struct  
__gnu_cxx::__per_type_pool_policy< _Tp, _PoolTp, _Thread >`

Policy for individual \_\_pool objects.

Definition at line 553 of file `mt_allocator.h`.

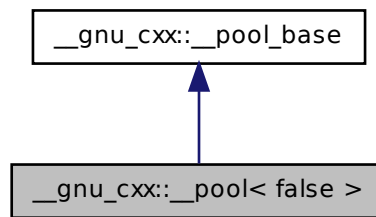
The documentation for this struct was generated from the following file:

- [mt\\_allocator.h](#)

## 5.9 \_\_gnu\_cxx::\_\_pool< false > Class Template Reference

Specialization for single thread.

Inheritance diagram for \_\_gnu\_cxx::\_\_pool< false >:



### Public Types

- typedef unsigned short int **\_Binmap\_type**

### Public Member Functions

- **\_\_pool** (const \_\_pool\_base::\_\_Tune &\_\_tune)
- void **\_M\_adjust\_freelist** (const \_Bin\_record &, \_Block\_record \*, size\_t)
- bool **\_M\_check\_threshold** (size\_t \_\_bytes)
- void **\_M\_destroy** () throw ()
- size\_t **\_M\_get\_align** ()
- const \_Bin\_record & **\_M\_get\_bin** (size\_t \_\_which)
- size\_t **\_M\_get\_binmap** (size\_t \_\_bytes)
- const \_Tune & **\_M\_get\_options** () const
- size\_t **\_M\_get\_thread\_id** ()
- void **\_M\_initialize\_once** ()
- void **\_M\_reclaim\_block** (char \*\_\_p, size\_t \_\_bytes) throw ()
- char \* **\_M\_reserve\_block** (size\_t \_\_bytes, const size\_t \_\_thread\_id)
- void **\_M\_set\_options** (\_Tune \_\_t)

### Protected Attributes

- `_Binmap_type * _M_binmap`
- `bool _M_init`
- `_Tune _M_options`

### 5.9.1 Detailed Description

`template<> class __gnu_cxx::__pool< false >`

Specialization for single thread.

Definition at line 194 of file `mt_allocator.h`.

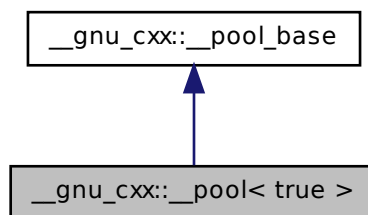
The documentation for this class was generated from the following file:

- [mt\\_allocator.h](#)

## 5.10 \_\_gnu\_cxx::\_\_pool< true > Class Template Reference

Specialization for thread enabled, via `gthreads.h`.

Inheritance diagram for `__gnu_cxx::__pool< true >`:



### Public Types

- `typedef unsigned short int _Binmap_type`

**Public Member Functions**

- `__pool` (const `__pool_base::__Tune &__tune`)
- `void _M_adjust_freelist` (const `_Bin_record &__bin`, `_Block_record *__block`, `size_t __thread_id`)
- `bool _M_check_threshold` (`size_t __bytes`)
- `void _M_destroy` () throw ()
- `void _M_destroy_thread_key` (`void *`) throw ()
- `size_t _M_get_align` ()
- `const _Bin_record & _M_get_bin` (`size_t __which`)
- `size_t _M_get_binmap` (`size_t __bytes`)
- `const _Tune & _M_get_options` () const
- `size_t _M_get_thread_id` ()
- `void _M_initialize` (`__destroy_handler`)
- `void _M_initialize_once` ()
- `void _M_reclaim_block` (`char *__p`, `size_t __bytes`) throw ()
- `char * _M_reserve_block` (`size_t __bytes`, const `size_t __thread_id`)
- `void _M_set_options` (`_Tune __t`)

**Protected Attributes**

- `_Binmap_type * _M_binmap`
- `bool _M_init`
- `_Tune _M_options`

**5.10.1 Detailed Description**

`template<> class __gnu_cxx::__pool< true >`

Specialization for thread enabled, via `gthreads.h`.

Definition at line 261 of file `mt_allocator.h`.

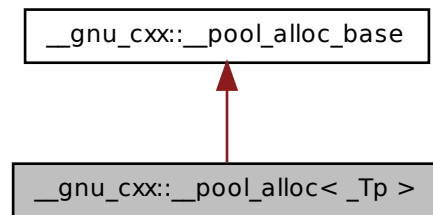
The documentation for this class was generated from the following file:

- [mt\\_allocator.h](#)

**5.11 `__gnu_cxx::__pool_alloc<_Tp>` Class Template Reference**

Allocator using a memory pool with a single lock.

Inheritance diagram for `__gnu_cxx::__pool_alloc<_Tp>`:



### Public Types

- `typedef const _Tp * const_pointer`
- `typedef const _Tp & const_reference`
- `typedef ptrdiff_t difference_type`
- `typedef _Tp * pointer`
- `typedef _Tp & reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

### Public Member Functions

- `__pool_alloc (const \_\_pool\_alloc &) throw ()`
- `template<typename _Tp1 >  
__pool_alloc (const \_\_pool\_alloc<_Tp1> &) throw ()`
- `pointer address (reference __x) const`
- `const_pointer address (const_reference __x) const`
- `pointer allocate (size_type __n, const void * = 0)`
- `template<typename... _Args>  
void construct (pointer __p, _Args &&... __args)`
- `void construct (pointer __p, const _Tp & __val)`
- `void deallocate (pointer __p, size_type __n)`
- `void destroy (pointer __p)`
- `size_type max_size () const throw ()`

**Private Types**

- enum { `_S_align` }
- enum { `_S_max_bytes` }
- enum { `_S_free_list_size` }

**Private Member Functions**

- `char * _M_allocate_chunk (size_t __n, int &__nobjs)`
- `_Obj *volatile * _M_get_free_list (size_t __bytes) throw ()`
- `__mutex & _M_get_mutex () throw ()`
- `void * _M_refill (size_t __n)`
- `size_t _M_round_up (size_t __bytes)`

**Static Private Attributes**

- static `char * _S_end_free`
- static `_Obj *volatile _S_free_list [_S_free_list_size]`
- static `size_t _S_heap_size`
- static `char * _S_start_free`

**5.11.1 Detailed Description**

**template<typename \_Tp> class `__gnu_cxx::__pool_alloc`< \_Tp >**

Allocator using a memory pool with a single lock.

Definition at line 124 of file `pool_allocator.h`.

The documentation for this class was generated from the following file:

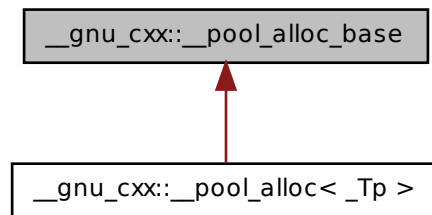
- [pool\\_allocator.h](#)

**5.12 `__gnu_cxx::__pool_alloc_base` Class Reference**

Base class for [\\_\\_pool\\_alloc](#).



Inheritance diagram for `__gnu_cxx::__pool_alloc_base`:



### Protected Types

- enum { **\_S\_align** }
- enum { **\_S\_max\_bytes** }
- enum { **\_S\_free\_list\_size** }

### Protected Member Functions

- `char * _M_allocate_chunk (size_t __n, int &__nobjs)`
- `_Obj *volatile * _M_get_free_list (size_t __bytes) throw ()`
- `__mutex & _M_get_mutex () throw ()`
- `void * _M_refill (size_t __n)`
- `size_t _M_round_up (size_t __bytes)`

### Static Protected Attributes

- `static char * _S_end_free`
- `static _Obj *volatile _S_free_list [_S_free_list_size]`
- `static size_t _S_heap_size`
- `static char * _S_start_free`

#### 5.12.1 Detailed Description

Base class for `__pool_alloc`. Uses various allocators to fulfill underlying requests (and makes as few requests as possible when in default high-speed pool mode).

Important implementation properties: 0. If globally mandated, then allocate objects from new 1. If the clients request an object of size  $> \_S\_max\_bytes$ , the resulting object will be obtained directly from new 2. In all other cases, we allocate an object of size exactly `\_S_round_up(requested_size)`. Thus the client has enough size information that we can return the object to the proper free list without permanently losing part of the object.

Definition at line 76 of file `pool_allocator.h`.

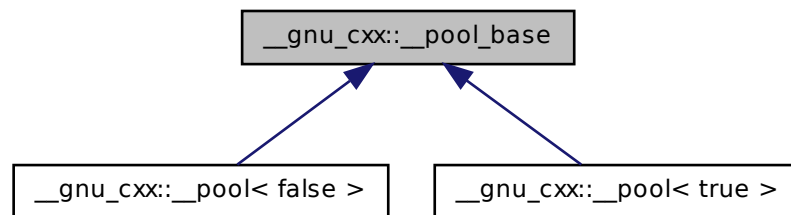
The documentation for this class was generated from the following file:

- [pool\\_allocator.h](#)

### 5.13 `__gnu_cxx::__pool_base` Struct Reference

Base class for pool object.

Inheritance diagram for `__gnu_cxx::__pool_base`:



#### Public Types

- typedef unsigned short int `_Binmap_type`

#### Public Member Functions

- `__pool_base` (const `_Tune` & `__options`)
- bool `_M_check_threshold` (size\_t `__bytes`)
- size\_t `_M_get_align` ()
- size\_t `_M_get_binmap` (size\_t `__bytes`)

## 5.14 `__gnu_cxx::__rc_string_base<_CharT, _Traits, _Alloc>` Class Template Reference 832

---

- `const _Tune & _M_get_options () const`
- `void _M_set_options (_Tune __t)`

### Protected Attributes

- `_Binmap_type * _M_binmap`
- `bool _M_init`
- `_Tune _M_options`

### 5.13.1 Detailed Description

Base class for pool object.

Definition at line 49 of file `mt_allocator.h`.

The documentation for this struct was generated from the following file:

- [mt\\_allocator.h](#)

## 5.14 `__gnu_cxx::__rc_string_base<_CharT, _Traits, _Alloc>` Class Template Reference

Inherits `__gnu_cxx::__vstring_utility<_CharT, _Traits, _Alloc>`.

### Public Types

- `typedef _Util_Base::_CharT_alloc_type _CharT_alloc_type`
- `typedef __vstring_utility<_CharT, _Traits, _Alloc> _Util_Base`
- `typedef _Alloc allocator_type`
- `typedef _CharT_alloc_type::size_type size_type`
- `typedef _Traits traits_type`
- `typedef _Traits::char_type value_type`

### Public Member Functions

- `__rc_string_base (const _Alloc &__a)`
- `__rc_string_base (const __rc_string_base &__rcs)`
- `__rc_string_base (__rc_string_base &&__rcs)`
- `__rc_string_base (size_type __n, _CharT __c, const _Alloc &__a)`
- `template<typename _InputIterator>  
__rc_string_base (_InputIterator __beg, _InputIterator __end, const _Alloc &__a)`

- `void _M_assign (const __rc_string_base &__rcs)`
- `size_type _M_capacity () const`
- `void _M_clear ()`
- `bool _M_compare (const __rc_string_base &) const`
- `template<>`  
`bool _M_compare (const __rc_string_base &__rcs) const`
- `template<>`  
`bool _M_compare (const __rc_string_base &__rcs) const`
- `_CharT * _M_data () const`
- `void _M_erase (size_type __pos, size_type __n)`
- `allocator_type & _M_get_allocator ()`
- `const allocator_type & _M_get_allocator () const`
- `bool _M_is_shared () const`
- `void _M_leak ()`
- `size_type _M_length () const`
- `size_type _M_max_size () const`
- `void _M_mutate (size_type __pos, size_type __len1, const _CharT * __s, size_type __len2)`
- `void _M_reserve (size_type __res)`
- `void _M_set_leaked ()`
- `void _M_set_length (size_type __n)`
- `void _M_swap (__rc_string_base &__rcs)`
- `template<typename _InIterator >`  
`_CharT * _S_construct (_InIterator __beg, _InIterator __end, const _Alloc & __a, std::forward_iterator_tag)`

#### Protected Types

- `typedef __gnu_cxx::__normal_iterator< const_pointer, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, __rc_string_base > > __const_rc_iterator`
- `typedef __gnu_cxx::__normal_iterator< const_pointer, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, __sso_string_base > > __const_sso_iterator`
- `typedef __gnu_cxx::__normal_iterator< pointer, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, __rc_string_base > > __rc_iterator`
- `typedef __gnu_cxx::__normal_iterator< pointer, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, __sso_string_base > > __sso_iterator`
- `typedef _CharT_alloc_type::const_pointer const_pointer`
- `typedef _CharT_alloc_type::difference_type difference_type`
- `typedef _CharT_alloc_type::pointer pointer`

### Static Protected Member Functions

- static void `_S_assign` (`_CharT *__d`, `size_type __n`, `_CharT __c`)
- static int `_S_compare` (`size_type __n1`, `size_type __n2`)
- static void `_S_copy` (`_CharT *__d`, `const _CharT *__s`, `size_type __n`)
- static void `_S_copy_chars` (`_CharT *__p`, `_CharT *__k1`, `_CharT *__k2`)
- static void `_S_copy_chars` (`_CharT *__p`, `__sso_iterator __k1`, `__sso_iterator __k2`)
- static void `_S_copy_chars` (`_CharT *__p`, `__rc_iterator __k1`, `__rc_iterator __k2`)
- static void `_S_copy_chars` (`_CharT *__p`, `__const_sso_iterator __k1`, `__const_sso_iterator __k2`)
- template<typename `_Iterator`>  
static void `_S_copy_chars` (`_CharT *__p`, `_Iterator __k1`, `_Iterator __k2`)
- static void `_S_copy_chars` (`_CharT *__p`, `__const_rc_iterator __k1`, `__const_rc_iterator __k2`)
- static void `_S_copy_chars` (`_CharT *__p`, `const _CharT *__k1`, `const _CharT *__k2`)
- static void `_S_move` (`_CharT *__d`, `const _CharT *__s`, `size_type __n`)

#### 5.14.1 Detailed Description

`template<typename _CharT, typename _Traits, typename _Alloc> class __gnu_cxx::__rc_string_base<_CharT, _Traits, _Alloc>`

Documentation? What's that? Nathan Myers <[ncm@cantrip.org](mailto:ncm@cantrip.org)>.

A string looks like this:

|                                                                             |                                                                                  |
|-----------------------------------------------------------------------------|----------------------------------------------------------------------------------|
| <pre> [__rc_string_base&lt;char_type&gt;] _M_dataplus _M_p -----&gt; </pre> | <pre> [_Rep] _M_length _M_capacity _M_refcount unnamed array of char_type </pre> |
|-----------------------------------------------------------------------------|----------------------------------------------------------------------------------|

Where the `_M_p` points to the first character in the string, and you cast it to a pointer-to-`_Rep` and subtract 1 to get a pointer to the header.

This approach has the enormous advantage that a string object requires only one allocation. All the ugliness is confined within a single pair of inline functions, which each compile to a single *add* instruction: `_Rep::_M_refdata()`, and `__rc_string_base::_M_rep()`; and the allocation function which gets a block of raw bytes and with room enough and constructs a `_Rep` object at the front.

The reason you want `_M_data` pointing to the character array and not the `_Rep` is so that the debugger can see the string contents. (Probably we should add a non-inline

member to get the `_Rep` for the debugger to use, so users can check the actual string length.)

Note that the `_Rep` object is a POD so that you can have a static *empty string* `_Rep` object already *constructed* before static constructors have run. The reference-count encoding is chosen so that a 0 indicates one reference, so you never try to destroy the empty-string `_Rep` object.

All but the last paragraph is considered pretty conventional for a C++ string implementation.

Definition at line 83 of file `rc_string_base.h`.

The documentation for this class was generated from the following file:

- [rc\\_string\\_base.h](#)

## 5.15 `__gnu_cxx::__scoped_lock` Class Reference

Scoped lock idiom.

### Public Types

- typedef `__mutex` `__mutex_type`

### Public Member Functions

- `__scoped_lock` (`__mutex_type` &`__name`)

#### 5.15.1 Detailed Description

Scoped lock idiom.

Definition at line 299 of file `concurrency.h`.

The documentation for this class was generated from the following file:

- [concurrency.h](#)

## 5.16 `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>` Class Template Reference

Template class [\\_\\_versa\\_string](#).

Data structure managing sequences of characters and character-like objects.

Inherits `_Base<_CharT, _Traits, _Alloc>`.

## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `__gnu_cxx::__normal_iterator< const_pointer, __versa_string >` **const\_iterator**
- typedef `_CharT_alloc_type::const_pointer` **const\_pointer**
- typedef `const value_type &` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_CharT_alloc_type::difference_type` **difference\_type**
- typedef `__gnu_cxx::__normal_iterator< pointer, __versa_string >` **iterator**
- typedef `_CharT_alloc_type::pointer` **pointer**
- typedef `value_type &` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `_CharT_alloc_type::size_type` **size\_type**
- typedef `_Traits` **traits\_type**
- typedef `_Traits::char_type` **value\_type**

## Public Member Functions

- `__versa_string ()`
- `__versa_string (const _Alloc &__a)`
- `__versa_string (__versa_string &&__str)`
- `__versa_string (const _CharT *__s, size_type __n, const _Alloc &__a=_Alloc())`
- `__versa_string (const _CharT *__s, const _Alloc &__a=_Alloc())`
- `__versa_string (std::initializer_list< _CharT > __l, const _Alloc &__a=_Alloc())`
- `__versa_string (size_type __n, _CharT __c, const _Alloc &__a=_Alloc())`
- `template<class _InputIterator >`  
`__versa_string (_InputIterator __beg, _InputIterator __end, const _Alloc &__a=_Alloc())`
- `__versa_string (const __versa_string &__str)`
- `__versa_string (const __versa_string &__str, size_type __pos, size_type __n=npos)`
- `__versa_string (const __versa_string &__str, size_type __pos, size_type __n, const _Alloc &__a)`
- `~__versa_string ()`
- `__versa_string & append (const __versa_string &__str)`
- `__versa_string & append (const __versa_string &__str, size_type __pos, size_type __n)`
- `__versa_string & append (const _CharT *__s, size_type __n)`
- `__versa_string & append (const _CharT *__s)`
- `__versa_string & append (size_type __n, _CharT __c)`
- `__versa_string & append (std::initializer_list< _CharT > __l)`

- `template<class _InputIterator >`  
`__versa_string & append` (`_InputIterator __first`, `_InputIterator __last`)
- `__versa_string & assign` (`const _CharT *__s`)
- `__versa_string & assign` (`size_type __n`, `_CharT __c`)
- `template<class _InputIterator >`  
`__versa_string & assign` (`_InputIterator __first`, `_InputIterator __last`)
- `__versa_string & assign` (`std::initializer_list<_CharT> __l`)
- `__versa_string & assign` (`const __versa_string &__str`)
- `__versa_string & assign` (`__versa_string &&__str`)
- `__versa_string & assign` (`const __versa_string &__str`, `size_type __pos`, `size_type __n`)
- `__versa_string & assign` (`const _CharT *__s`, `size_type __n`)
- `const_reference at` (`size_type __n`) `const`
- `reference at` (`size_type __n`)
- `reference back` ()
- `const_reference back` () `const`
- `iterator begin` ()
- `const_iterator begin` () `const`
- `const _CharT * c_str` () `const`
- `size_type capacity` () `const`
- `const_iterator cbegin` () `const`
- `const_iterator cend` () `const`
- `void clear` ()
- `int compare` (`const _CharT *__s`) `const`
- `int compare` (`size_type __pos`, `size_type __n1`, `const _CharT *__s`) `const`
- `int compare` (`size_type __pos`, `size_type __n1`, `const _CharT *__s`, `size_type __n2`) `const`
- `int compare` (`size_type __pos1`, `size_type __n1`, `const __versa_string &__str`, `size_type __pos2`, `size_type __n2`) `const`
- `int compare` (`size_type __pos`, `size_type __n`, `const __versa_string &__str`) `const`
- `int compare` (`const __versa_string &__str`) `const`
- `size_type copy` (`_CharT *__s`, `size_type __n`, `size_type __pos=0`) `const`
- `const_reverse_iterator crbegin` () `const`
- `const_reverse_iterator crend` () `const`
- `const _CharT * data` () `const`
- `bool empty` () `const`
- `iterator end` ()
- `const_iterator end` () `const`
- `__versa_string & erase` (`size_type __pos=0`, `size_type __n=npos`)
- `iterator erase` (`iterator __position`)
- `iterator erase` (`iterator __first`, `iterator __last`)
- `size_type find` (`_CharT __c`, `size_type __pos=0`) `const`
- `size_type find` (`const _CharT *__s`, `size_type __pos`, `size_type __n`) `const`



- `size_type find` (const `__versa_string` &\_\_str, `size_type` \_\_pos=0) const
- `size_type find` (const `_CharT` \*\_\_s, `size_type` \_\_pos=0) const
- `size_type find_first_not_of` (const `_CharT` \*\_\_s, `size_type` \_\_pos, `size_type` \_\_n) const
- `size_type find_first_not_of` (`_CharT` \_\_c, `size_type` \_\_pos=0) const
- `size_type find_first_not_of` (const `__versa_string` &\_\_str, `size_type` \_\_pos=0) const
- `size_type find_first_not_of` (const `_CharT` \*\_\_s, `size_type` \_\_pos=0) const
- `size_type find_first_of` (`_CharT` \_\_c, `size_type` \_\_pos=0) const
- `size_type find_first_of` (const `_CharT` \*\_\_s, `size_type` \_\_pos, `size_type` \_\_n) const
- `size_type find_first_of` (const `_CharT` \*\_\_s, `size_type` \_\_pos=0) const
- `size_type find_first_of` (const `__versa_string` &\_\_str, `size_type` \_\_pos=0) const
- `size_type find_last_not_of` (`_CharT` \_\_c, `size_type` \_\_pos=`npos`) const
- `size_type find_last_not_of` (const `_CharT` \*\_\_s, `size_type` \_\_pos, `size_type` \_\_n) const
- `size_type find_last_not_of` (const `__versa_string` &\_\_str, `size_type` \_\_pos=`npos`) const
- `size_type find_last_not_of` (const `_CharT` \*\_\_s, `size_type` \_\_pos=`npos`) const
- `size_type find_last_of` (`_CharT` \_\_c, `size_type` \_\_pos=`npos`) const
- `size_type find_last_of` (const `__versa_string` &\_\_str, `size_type` \_\_pos=`npos`) const
- `size_type find_last_of` (const `_CharT` \*\_\_s, `size_type` \_\_pos=`npos`) const
- `size_type find_last_of` (const `_CharT` \*\_\_s, `size_type` \_\_pos, `size_type` \_\_n) const
- reference `front` ()
- const\_reference `front` () const
- allocator\_type `get_allocator` () const
- void `insert` (iterator \_\_p, `size_type` \_\_n, `_CharT` \_\_c)
- template<class `_InputIterator` >  
     void `insert` (iterator \_\_p, `_InputIterator` \_\_beg, `_InputIterator` \_\_end)
- void `insert` (iterator \_\_p, `std::initializer_list`< `_CharT` > \_\_l)
- `__versa_string` & `insert` (`size_type` \_\_pos1, const `__versa_string` &\_\_str, `size_type` \_\_pos2, `size_type` \_\_n)
- `__versa_string` & `insert` (`size_type` \_\_pos, const `_CharT` \*\_\_s, `size_type` \_\_n)
- `__versa_string` & `insert` (`size_type` \_\_pos, const `_CharT` \*\_\_s)
- iterator `insert` (iterator \_\_p, `_CharT` \_\_c)
- `__versa_string` & `insert` (`size_type` \_\_pos, `size_type` \_\_n, `_CharT` \_\_c)
- `__versa_string` & `insert` (`size_type` \_\_pos1, const `__versa_string` &\_\_str)
- `size_type length` () const
- `size_type max_size` () const
- `__versa_string` & `operator+=` (`std::initializer_list`< `_CharT` > \_\_l)
- `__versa_string` & `operator+=` (const `__versa_string` &\_\_str)

- `__versa_string & operator+= (_CharT __c)`
- `__versa_string & operator+= (const _CharT *__s)`
- `__versa_string & operator= (__versa_string &&__str)`
- `__versa_string & operator= (const _CharT *__s)`
- `__versa_string & operator= (_CharT __c)`
- `__versa_string & operator= (std::initializer_list<_CharT> __l)`
- `__versa_string & operator= (const __versa_string &__str)`
- `const_reference operator[] (size_type __pos) const`
- `reference operator[] (size_type __pos)`
- `void push_back (_CharT __c)`
- `reverse_iterator rbegin ()`
- `const_reverse_iterator rbegin () const`
- `reverse_iterator rend ()`
- `const_reverse_iterator rend () const`
- `__versa_string & replace (iterator __i1, iterator __i2, size_type __n, _CharT __c)`
- `__versa_string & replace (iterator __i1, iterator __i2, const __versa_string &__str)`
- `__versa_string & replace (size_type __pos, size_type __n1, size_type __n2, _CharT __c)`
- `__versa_string & replace (size_type __pos, size_type __n1, const _CharT *__s)`
- `__versa_string & replace (size_type __pos, size_type __n, const __versa_string &__str)`
- `__versa_string & replace (iterator __i1, iterator __i2, std::initializer_list<_CharT> __l)`
- `template<class _InputIterator>  
__versa_string & replace (iterator __i1, iterator __i2, _InputIterator __k1, _InputIterator __k2)`
- `__versa_string & replace (iterator __i1, iterator __i2, iterator __k1, iterator __k2)`
- `__versa_string & replace (iterator __i1, iterator __i2, const _CharT *__s)`
- `__versa_string & replace (iterator __i1, iterator __i2, const_iterator __k1, const_iterator __k2)`
- `__versa_string & replace (size_type __pos1, size_type __n1, const __versa_string &__str, size_type __pos2, size_type __n2)`
- `__versa_string & replace (size_type __pos, size_type __n1, const _CharT *__s, size_type __n2)`
- `__versa_string & replace (iterator __i1, iterator __i2, const _CharT *__k1, const _CharT *__k2)`
- `__versa_string & replace (iterator __i1, iterator __i2, _CharT *__k1, _CharT *__k2)`
- `__versa_string & replace (iterator __i1, iterator __i2, const _CharT *__s, size_type __n)`

- void `reserve` (size\_type \_\_res\_arg=0)
- void `resize` (size\_type \_\_n)
- void `resize` (size\_type \_\_n, \_CharT \_\_c)
- size\_type `rfind` (const \_CharT \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const
- size\_type `rfind` (const `__versa_string` &\_\_str, size\_type \_\_pos=`npos`) const
- size\_type `rfind` (const \_CharT \*\_\_s, size\_type \_\_pos=`npos`) const
- size\_type `rfind` (\_CharT \_\_c, size\_type \_\_pos=`npos`) const
- void `shrink_to_fit` ()
- size\_type `size` () const
- `__versa_string` `substr` (size\_type \_\_pos=0, size\_type \_\_n=`npos`) const
- void `swap` (`__versa_string` &\_\_s)

#### Static Public Attributes

- static const size\_type `npos`

#### 5.16.1 Detailed Description

`template<typename _CharT, typename _Traits, typename _Alloc, template<typename, typename, typename> class _Base> class __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>`

Template class `__versa_string`.

Data structure managing sequences of characters and character-like objects.

Definition at line 54 of file `vstring.h`.

#### 5.16.2 Constructor & Destructor Documentation

**5.16.2.1** `template<typename _CharT, typename _Traits, typename _Alloc, template<typename, typename, typename> class _Base> __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::__versa_string( ) [inline]`

Default constructor creates an empty string.

Definition at line 132 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::substr()`.

**5.16.2.2** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base  
>::__versa_string( const _Alloc & __a ) [inline, explicit]`

Construct an empty string using allocator *a*.

Definition at line 139 of file `vstring.h`.

**5.16.2.3** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base  
>::__versa_string( const __versa_string< _CharT, _Traits, _Alloc,  
_Base > & __str ) [inline]`

Construct string with copy of value of *str*.

#### Parameters

`__str` Source string.

Definition at line 147 of file `vstring.h`.

**5.16.2.4** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base  
>::__versa_string( __versa_string< _CharT, _Traits, _Alloc, _Base >  
&& __str ) [inline]`

String move constructor.

#### Parameters

`__str` Source string.

The newly-constructed string contains the exact contents of *str*. The contents of *str* are a valid, but unspecified string.

Definition at line 159 of file `vstring.h`.

```
5.16.2.5 template<typename _CharT, typename _Traits, typename _Alloc,
 template< typename, typename, typename > class _Base>
 __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
 >::__versa_string (std::initializer_list<_CharT> __l, const _Alloc
 & __a = _Alloc()) [inline]
```

Construct string from an initializer list.

#### Parameters

`__l` [std::initializer\\_list](#) of characters.  
`__a` Allocator to use (default is default allocator).

Definition at line 167 of file `vstring.h`.

```
5.16.2.6 template<typename _CharT, typename _Traits, typename _Alloc,
 template< typename, typename, typename > class _Base>
 __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
 >::__versa_string (const __versa_string< _CharT, _Traits, _Alloc,
 _Base > & __str, size_type __pos, size_type __n = npos)
 [inline]
```

Construct string as copy of a substring.

#### Parameters

`__str` Source string.  
`__pos` Index of first character to copy from.  
`__n` Number of characters to copy (default remainder).

Definition at line 178 of file `vstring.h`.

```
5.16.2.7 template<typename _CharT, typename _Traits, typename _Alloc,
 template< typename, typename, typename > class _Base>
 __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
 >::__versa_string (const __versa_string< _CharT, _Traits, _Alloc,
 _Base > & __str, size_type __pos, size_type __n, const _Alloc & __a
) [inline]
```

Construct string as copy of a substring.

**Parameters**

`__str` Source string.  
`__pos` Index of first character to copy from.  
`__n` Number of characters to copy.  
`__a` Allocator to use.

Definition at line 193 of file `vstring.h`.

```
5.16.2.8 template<typename _CharT, typename _Traits, typename _Alloc,
 template< typename, typename, typename > class _Base>
 __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
 >::__versa_string(const _CharT * __s, size_type __n, const _Alloc
 & __a = _Alloc()) [inline]
```

Construct string initialized by a character array.

**Parameters**

`__s` Source character array.  
`__n` Number of characters to copy.  
`__a` Allocator to use (default is default allocator).

NB: `__s` must have at least `__n` characters, `'\0'` has no special meaning.

Definition at line 210 of file `vstring.h`.

```
5.16.2.9 template<typename _CharT, typename _Traits, typename _Alloc,
 template< typename, typename, typename > class _Base>
 __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
 >::__versa_string(const _CharT * __s, const _Alloc & __a =
 _Alloc()) [inline]
```

Construct string as copy of a C string.

**Parameters**

`__s` Source C string.  
`__a` Allocator to use (default is default allocator).

Definition at line 219 of file `vstring.h`.

**5.16.2.10** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base  
>::__versa_string ( size_type __n, _CharT __c, const _Alloc & __a  
= _Alloc() ) [inline]`

Construct string as multiple characters.

**Parameters**

- `__n` Number of characters.
- `__c` Character to use.
- `__a` Allocator to use (default is default allocator).

Definition at line 229 of file `vstring.h`.

**5.16.2.11** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
template<class _InputIterator > __gnu_cxx::__versa_string<  
_CharT, _Traits, _Alloc, _Base >::__versa_string ( _InputIterator  
__beg, _InputIterator __end, const _Alloc & __a = _Alloc() )  
[inline]`

Construct string as copy of a range.

**Parameters**

- `__beg` Start of range.
- `__end` End of range.
- `__a` Allocator to use (default is default allocator).

Definition at line 239 of file `vstring.h`.

**5.16.2.12** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base  
>::~~__versa_string ( ) [inline]`

Destroy the string instance.

Definition at line 246 of file `vstring.h`.

### 5.16.3 Member Function Documentation

**5.16.3.1** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc,  
_Base>::append ( const __versa_string<_CharT, _Traits, _Alloc,  
_Base> & __str ) [inline]`

Append a string to this string.

#### Parameters

`__str` The string to append.

#### Returns

Reference to this string.

Definition at line 678 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `std::getline()`, `__gnu_cxx::operator+()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::operator+=()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::resize()`.

**5.16.3.2** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc,  
_Base>::append ( const __versa_string<_CharT, _Traits, _Alloc,  
_Base> & __str, size_type __pos, size_type __n ) [inline]`

Append a substring.

#### Parameters

`__str` The string to append.

`__pos` Index of the first character of `str` to append.

`__n` The number of characters to append.

#### Returns

Reference to this string.



### Exceptions

[\*`std::out\_of\_range`\*](#) if *pos* is not a valid index.

This function appends *n* characters from *str* starting at *pos* to this string. If *n* is larger than the number of available characters in *str*, the remainder of *str* is appended.

Definition at line 695 of file `vstring.h`.

**5.16.3.3** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc,  
_Base>::append ( const _CharT * __s, size_type __n ) [inline]`

Append a C substring.

### Parameters

*\_\_s* The C string to append.  
*\_\_n* The number of characters to append.

### Returns

Reference to this string.

Definition at line 707 of file `vstring.h`.

**5.16.3.4** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc,  
_Base>::append ( const _CharT * __s ) [inline]`

Append a C string.

### Parameters

*\_\_s* The C string to append.

### Returns

Reference to this string.

Definition at line 720 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::length()`.

**5.16.3.5** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc,  
_Base>::append ( size_type __n, _CharT __c ) [inline]`

Append multiple characters.

**Parameters**

`__n` The number of characters to append.  
`__c` The character to use.

**Returns**

Reference to this string.

Appends `n` copies of `c` to this string.

Definition at line 737 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**5.16.3.6** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc,  
_Base>::append ( std::initializer_list<_CharT> __l ) [inline]`

Append an `initializer_list` of characters.

**Parameters**

`__l` The `initializer_list` of characters to append.

**Returns**

Reference to this string.

Definition at line 747 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append()`.

**5.16.3.7** `template<typename _CharT, typename _Traits, typename  
_Alloc, template< typename, typename, typename > class  
_Base> template<class _InputIterator > __versa_string&  
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base  
>::append ( _InputIterator __first, _InputIterator __last )  
[inline]`

Append a range of characters.

**Parameters**

`__first` Iterator referencing the first character to append.

`__last` Iterator marking the end of the range.

**Returns**

Reference to this string.

Appends characters in the range [first,last) to this string.

Definition at line 761 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

**5.16.3.8** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,  
_Base >::assign ( const __versa_string< _CharT, _Traits, _Alloc,  
_Base > & __str ) [inline]`

Set value to contents of another string.

**Parameters**

`__str` Source string to use.

**Returns**

Reference to this string.

Definition at line 784 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::operator=()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::push_back()`.

**5.16.3.9** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,  
_Base >::assign ( __versa_string< _CharT, _Traits, _Alloc, _Base >  
&& __str ) [inline]`

Set value to contents of another string.

#### Parameters

`__str` Source string to use.

#### Returns

Reference to this string.

This function sets this string to the exact contents of `__str`. `__str` is a valid, but unspecified string.

Definition at line 800 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::swap()`.

**5.16.3.10** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,  
_Alloc, _Base >::assign ( const __versa_string< _CharT, _Traits,  
_Alloc, _Base > & __str, size_type __pos, size_type __n )  
[inline]`

Set value to a substring of a string.

#### Parameters

`__str` The string to use.

`__pos` Index of the first character of str.

`__n` Number of characters to use.

#### Returns

Reference to this string.

#### Exceptions

[\*`std::out\_of\_range`\*](#) if `__pos` is not a valid index.

This function sets this string to the substring of `__str` consisting of `__n` characters at `__pos`. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is used.

Definition at line 821 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**5.16.3.11** `template<typename _CharT, typename _Traits, typename _Alloc,`  
`template< typename, typename, typename > class _Base>`  
`__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits,`  
`_Alloc, _Base>::assign ( const _CharT * __s, size_type __n )`  
`[inline]`

Set value to a C substring.

#### Parameters

`__s` The C string to use.

`__n` Number of characters to use.

#### Returns

Reference to this string.

This function sets the value of this string to the first `__n` characters of `__s`. If `__n` is larger than the number of available characters in `__s`, the remainder of `__s` is used.

Definition at line 838 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**5.16.3.12** `template<typename _CharT, typename _Traits, typename _Alloc,`  
`template< typename, typename, typename > class _Base>`  
`__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits,`  
`_Alloc, _Base>::assign ( const _CharT * __s ) [inline]`

Set value to contents of a C string.

#### Parameters

`__s` The C string to use.

#### Returns

Reference to this string.

## 5.16 `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>` Class Template Reference 851

---

This function sets the value of this string to the value of `__s`. The data is copied, so there is no dependence on `__s` once the function returns.

Definition at line 854 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**5.16.3.13** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::assign ( size_type __n, _CharT __c ) [inline]`

Set value to multiple characters.

### Parameters

`__n` Length of the resulting string.

`__c` The character to use.

### Returns

Reference to this string.

This function sets the value of this string to `__n` copies of character `__c`.

Definition at line 871 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**5.16.3.14** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> template<class _InputIterator > __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::assign ( _InputIterator __first, _InputIterator __last ) [inline]`

Set value to a range of characters.

### Parameters

`__first` Iterator referencing the first character to append.

`__last` Iterator marking the end of the range.

### Returns

Reference to this string.

Sets value of string to characters in the range [first,last).

Definition at line 885 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

**5.16.3.15** `template<typename _CharT, typename _Traits, typename _Alloc,`  
`template< typename, typename, typename > class _Base>`  
`__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits,`  
`_Alloc, _Base >::assign ( std::initializer_list<_CharT > __l )`  
`[inline]`

Set value to an `initializer_list` of characters.

#### Parameters

`__l` The `initializer_list` of characters to assign.

#### Returns

Reference to this string.

Definition at line 895 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::assign()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::assign()`.

**5.16.3.16** `template<typename _CharT, typename _Traits, typename _Alloc,`  
`template< typename, typename, typename > class _Base>`  
`const_reference __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc,`  
`_Base >::at ( size_type __n ) const [inline]`

Provides access to the data contained in the string.

#### Parameters

`__n` The index of the character to access.

#### Returns

Read-only (const) reference to the character.

#### Exceptions

*[`std::out\_of\_range`](#)* If `__n` is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails.

Definition at line 569 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**5.16.3.17** `template<typename _CharT, typename _Traits, typename _Alloc,`  
`template< typename, typename, typename > class _Base> reference`  
`__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::at (`  
`size_type __n ) [inline]`

Provides access to the data contained in the string.

#### Parameters

`__n` The index of the character to access.

#### Returns

Read/write reference to the character.

#### Exceptions

*`std::out_of_range`* If `__n` is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails. Success results in unsharing the string.

Definition at line 588 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**5.16.3.18** `template<typename _CharT, typename _Traits, typename _Alloc,`  
`template< typename, typename, typename > class _Base>`  
`const_reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,`  
`_Base >::back ( ) const [inline]`

Returns a read-only (constant) reference to the data at the last element of the string.

Definition at line 626 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::operator[]()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.



**5.16.3.19** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base> reference  
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::back  
( ) [inline]`

Returns a read/write reference to the data at the last element of the string.

Definition at line 618 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base  
>::operator[]()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base  
>::size()`.

**5.16.3.20** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base> iterator  
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::begin  
( ) [inline]`

Returns a read/write iterator that points to the first character in the string. Unshares the string.

Definition at line 312 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base  
>::crend()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base  
>::rend()`.

**5.16.3.21** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,  
_Base >::begin ( ) const [inline]`

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 323 of file `vstring.h`.

**5.16.3.22** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base> const  
_CharT* __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base  
>::c_str ( ) const [inline]`

Return const pointer to null-terminated contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1487 of file `vstring.h`.

**5.16.3.23** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base> size_type  
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base  
>::capacity ( ) const [inline]`

Returns the total number of characters that the string can hold before needing to allocate more memory.

Definition at line 480 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::push_back()`.

**5.16.3.24** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,  
_Base >::cbegin ( ) const [inline]`

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 387 of file `vstring.h`.

**5.16.3.25** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,  
_Base >::cend ( ) const [inline]`

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 395 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**5.16.3.26** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base> void  
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::clear  
( ) [inline]`

Erases the string, making it empty.

Definition at line 508 of file `vstring.h`.

**5.16.3.27** `template<typename _CharT, typename _Traits, typename _Alloc,  
 template< typename, typename, typename > class _Base> int  
 __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base  
 >::compare ( const __versa_string< _CharT, _Traits, _Alloc, _Base  
 > & __str ) const [inline]`

Compare to a string.

#### Parameters

`__str` String to compare against.

#### Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before `__str`, 0 if their values are equivalent, or > 0 if this string is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and `str.size()`. The function then compares the two strings by calling `traits::compare(data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 1906 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::data()`, `std::min()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`, `__gnu_cxx::operator<()`, `__gnu_cxx::operator<=()`, `__gnu_cxx::operator==()`, `__gnu_cxx::operator>()`, and `__gnu_cxx::operator>=()`.

**5.16.3.28** `template<typename _CharT, typename _Traits, typename _Alloc,  
 template< typename, typename, typename > class _Base> int  
 __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base  
 >::compare ( size_type __pos, size_type __n, const __versa_string<  
 _CharT, _Traits, _Alloc, _Base > & __str ) const`

Compare substring to a string.

#### Parameters

`__pos` Index of first character of substring.

`__n` Number of characters in substring.

`__str` String to compare against.

### Returns

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Form the substring of this string from the `__n` characters starting at `__pos`. Returns an integer  $< 0$  if the substring is ordered before `__str`,  $0$  if their values are equivalent, or  $> 0$  if the substring is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__str.size()`. The function then compares the two strings by calling `traits::compare(substring.data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 460 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::data()`, `std::min()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**5.16.3.29** `template<typename _CharT, typename _Traits, typename _Alloc, template<typename, typename, typename> class _Base> int __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare ( size_type __pos1, size_type __n1, const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str, size_type __pos2, size_type __n2 ) const`

Compare substring to a substring.

### Parameters

- `__pos1` Index of first character of substring.
- `__n1` Number of characters in substring.
- `__str` String to compare against.
- `__pos2` Index of first character of substring of `str`.
- `__n2` Number of characters in substring of `str`.

### Returns

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Form the substring of this string from the `__n1` characters starting at `__pos1`. Form the substring of `__str` from the `__n2` characters starting at `__pos2`. Returns an integer  $< 0$  if this substring is ordered before the substring of `__str`,  $0$  if their values are equivalent, or  $> 0$  if this substring is ordered after the substring of `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of

the lengths of the substrings. The function then compares the two strings by calling `traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 477 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::data()`, and `std::min()`.

**5.16.3.30** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> int __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare ( const _CharT * __s ) const`

Compare to a C string.

#### Parameters

`__s` C string to compare against.

#### Returns

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Returns an integer  $< 0$  if this string is ordered before `__s`,  $0$  if their values are equivalent, or  $> 0$  if this string is ordered after `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and the length of a string constructed from `__s`. The function then compares the two strings by calling `traits::compare(data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 496 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::length()`, `std::min()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**5.16.3.31** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> int __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare ( size_type __pos, size_type __n1, const _CharT * __s ) const`

Compare substring to a C string.

#### Parameters

`__pos` Index of first character of substring.  
`__n1` Number of characters in substring.  
`__s` C string to compare against.

#### Returns

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Form the substring of this string from the `__n1` characters starting at `__pos`. Returns an integer  $< 0$  if the substring is ordered before `__s`,  $0$  if their values are equivalent, or  $> 0$  if the substring is ordered after `__s`. Determines the effective length `r1en` of the strings to compare as the smallest of the length of the substring and the length of a string constructed from `__s`. The function then compares the two string by calling `traits::compare(substring.data(),s,r1en)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 512 of file `vstring.tcc`.

References `std::min()`.

**5.16.3.32** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> int __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare( size_type __pos, size_type __n1, const _CharT* __s, size_type __n2 ) const`

Compare substring against a character array.

#### Parameters

`__pos1` Index of first character of substring.  
`__n1` Number of characters in substring.  
`__s` character array to compare against.  
`__n2` Number of characters of `s`.

#### Returns

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Form the substring of this string from the `__n1` characters starting at `__pos1`. Form a string from the first `__n2` characters of `__s`. Returns an integer  $< 0$  if this substring

is ordered before the string from `__s`, 0 if their values are equivalent, or  $> 0$  if this substring is ordered after the string from `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__n2`. The function then compares the two strings by calling `traits::compare(substring.data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

NB: `s` must have at least `n2` characters, `\0` has no special meaning.

Definition at line 529 of file `vstring.tcc`.

References `std::min()`.

**5.16.3.33** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::copy ( _CharT * __s, size_type __n, size_type __pos = 0 ) const`

Copy substring into C string.

#### Parameters

`__s` C string to copy value into.

`__n` Number of characters to copy.

`__pos` Index of first character to copy.

#### Returns

Number of characters actually copied

#### Exceptions

[`std::out\_of\_range`](#) If `pos > size()`.

Copies up to `__n` characters starting at `__pos` into the C string `s`. If `__pos` is greater than `size()`, `out_of_range` is thrown.

Definition at line 255 of file `vstring.tcc`.

**5.16.3.34** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::crbegin ( ) const [inline]`

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 404 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::end()`.

**5.16.3.35** `template<typename _CharT, typename _Traits, typename _Alloc,`  
`template< typename, typename, typename > class _Base>`  
`const reverse_iterator __gnu_cxx::__versa_string<_CharT, _Traits,`  
`_Alloc, _Base>::crend ( ) const [inline]`

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 413 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::begin()`.

**5.16.3.36** `template<typename _CharT, typename _Traits, typename _Alloc,`  
`template< typename, typename, typename > class _Base> const`  
`_CharT* __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base`  
`>::data ( ) const [inline]`

Return const pointer to contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1497 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_not_of()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_of()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_not_of()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_of()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::rfind()`.

**5.16.3.37** `template<typename _CharT, typename _Traits, typename _Alloc,`  
`template< typename, typename, typename > class _Base> bool`  
`__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base`  
`>::empty ( ) const [inline]`

Returns true if the string is empty. Equivalent to `*this == ""`.

Definition at line 516 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.



**5.16.3.38** `template<typename _CharT, typename _Traits, typename _Alloc,  
 template< typename, typename, typename > class _Base>  
 const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,  
 _Base >::end ( ) const [inline]`

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 342 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

**5.16.3.39** `template<typename _CharT, typename _Traits, typename _Alloc,  
 template< typename, typename, typename > class _Base> iterator  
 __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::end ( ) [inline]`

Returns a read/write iterator that points one past the last character in the string. Unshares the string.

Definition at line 331 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::crbegin()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rbegin()`.

**5.16.3.40** `template<typename _CharT, typename _Traits, typename _Alloc,  
 template< typename, typename, typename > class _Base>  
 __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,  
 _Alloc, _Base >::erase ( size_type __pos = 0, size_type __n = npos ) [inline]`

Remove characters.

#### Parameters

`__pos` Index of first character to remove (default 0).

`__n` Number of characters to remove (default remainder).

#### Returns

Reference to this string.

### Exceptions

*`std::out_of_range`* If `__pos` is beyond the end of this string.

Removes `__n` characters from this string starting at `__pos`. The length of the string is reduced by `__n`. If there are `< __n` characters to remove, the remainder of the string is truncated. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1090 of file `vstring.h`.

Referenced by `std::getline()`.

**5.16.3.41** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base> iterator  
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::erase  
( iterator __position ) [inline]`

Remove one character.

### Parameters

`__position` Iterator referencing the character to remove.

### Returns

iterator referencing same location after removal.

Removes the character at `__position` from this string. The value of the string doesn't change if an error is thrown.

Definition at line 1106 of file `vstring.h`.

**5.16.3.42** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base> iterator  
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::erase  
( iterator __first, iterator __last ) [inline]`

Remove a range of characters.

### Parameters

`__first` Iterator referencing the first character to remove.

`__last` Iterator referencing the end of the range.

### Returns

Iterator referencing location of first after removal.

Removes the characters in the range [first,last) from this string. The value of the string doesn't change if an error is thrown.

Definition at line 1127 of file `vstring.h`.

**5.16.3.43** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find ( const _CharT * __s, size_type __pos, size_type __n ) const`

Find position of a C substring.

### Parameters

`__s` C string to locate.  
`__pos` Index of character to search from.  
`__n` Number of characters from `__s` to search for.

### Returns

Index of start of first occurrence.

Starting from `__pos`, searches forward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 270 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::npos`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_of()`.

**5.16.3.44** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find ( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str, size_type __pos = 0 ) const [inline]`

Find position of a string.

**Parameters**

`__str` String to locate.  
`__pos` Index of character to search from (default 0).

**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1533 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::data()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find()`.

**5.16.3.45** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base> size_type  
__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find (  
const _CharT * __s, size_type __pos = 0 ) const [inline]`

Find position of a C string.

**Parameters**

`__s` C string to locate.  
`__pos` Index of character to search from (default 0).

**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1547 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find()`.

**5.16.3.46** `template<typename _CharT, typename _Traits, typename _Alloc  
, template< typename, typename, typename > class _Base>  
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type  
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find (  
_CharT __c, size_type __pos = 0 ) const`

Find position of a character.

#### Parameters

`__c` Character to locate.  
`__pos` Index of character to search from (default 0).

#### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 294 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::npos`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

**5.16.3.47** `template<typename _CharT, typename _Traits, typename _Alloc  
, template< typename, typename, typename > class _Base>  
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type  
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base  
>::find_first_not_of ( const _CharT * __s, size_type __pos,  
size_type __n ) const`

Find position of a character not in C substring.

#### Parameters

`__s` C string containing characters to avoid.  
`__pos` Index of character to search from.  
`__n` Number of characters from `s` to consider.

#### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 392 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::npos`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**5.16.3.48** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_not_of ( _CharT __c, size_type __pos = 0 ) const`

Find position of a different character.

#### Parameters

`__c` Character to avoid.

`__pos` Index of character to search from (default 0).

#### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 405 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::npos`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**5.16.3.49** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_not_of ( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str, size_type __pos = 0 ) const [inline]`

Find position of a character not in string.

#### Parameters

`__str` String containing characters to avoid.

`__pos` Index of character to search from (default 0).

### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1761 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::data()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_not_of()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_not_of()`.

```
5.16.3.50 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> size_type
__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base
>::find_first_not_of(const _CharT * __s, size_type __pos = 0)
const [inline]
```

Find position of a character not in C string.

### Parameters

`__s` C string containing characters to avoid.

`__pos` Index of character to search from (default 0).

### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1791 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_not_of()`.

**5.16.3.51** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base> size_type  
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base  
>::find_first_of ( const __versa_string< _CharT, _Traits, _Alloc,  
_Base > & __str, size_type __pos = 0 ) const [inline]`

Find position of a character of string.

#### Parameters

`__str` String containing characters to locate.  
`__pos` Index of character to search from (default 0).

#### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1636 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::data()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_of()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_of()`.

**5.16.3.52** `template<typename _CharT, typename _Traits, typename _Alloc  
, template< typename, typename, typename > class _Base>  
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type  
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base  
>::find_first_of ( const _CharT * __s, size_type __pos, size_type  
__n ) const`

Find position of a character of C substring.

#### Parameters

`__s` String containing characters to locate.  
`__pos` Index of character to search from.  
`__n` Number of characters from `s` to search for.



### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 353 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::npos`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**5.16.3.53** `template<typename _CharT, typename _Traits, typename _Alloc,`  
`template< typename, typename, typename > class _Base> size_type`  
`__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base`  
`>::find_first_of ( const _CharT * __s, size_type __pos = 0 ) const`  
`[inline]`

Find position of a character of C string.

### Parameters

`__s` String containing characters to locate.

`__pos` Index of character to search from (default 0).

### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1665 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_of()`.

**5.16.3.54** `template<typename _CharT, typename _Traits, typename _Alloc,`  
`template< typename, typename, typename > class _Base> size_type`  
`__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base`  
`>::find_first_of ( _CharT __c, size_type __pos = 0 ) const`  
`[inline]`

Find position of a character.

### Parameters

- `__c` Character to locate.
- `__pos` Index of character to search from (default 0).

### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for the character `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `find(c, pos)`.

Definition at line 1684 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find()`.

**5.16.3.55** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_not_of ( _CharT __c, size_type __pos = npos ) const`

Find last position of a different character.

### Parameters

- `__c` Character to avoid.
- `__pos` Index of character to search back from (default end).

### Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 439 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::npos`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**5.16.3.56** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base> size_type  
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base  
>::find_last_not_of ( const __versa_string< _CharT, _Traits, _Alloc,  
_Base > & __str, size_type __pos = npos ) const [inline]`

Find last position of a character not in string.

#### Parameters

`__str` String containing characters to avoid.

`__pos` Index of character to search back from (default end).

#### Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1822 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::data()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_not_of()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_not_of()`.

**5.16.3.57** `template<typename _CharT, typename _Traits, typename _Alloc  
, template< typename, typename, typename > class _Base>  
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type  
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base  
>::find_last_not_of ( const _CharT * __s, size_type __pos,  
size_type __n ) const`

Find last position of a character not in C substring.

#### Parameters

`__s` C string containing characters to avoid.

`__pos` Index of character to search back from.

`__n` Number of characters from `s` to consider.

### Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 417 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::npos`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

```
5.16.3.58 template<typename _CharT, typename _Traits, typename _Alloc,
 template< typename, typename, typename > class _Base> size_type
 __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
 >::find_last_not_of(const _CharT * __s, size_type __pos = npos)
 const [inline]
```

Find last position of a character not in C string.

### Parameters

`__s` C string containing characters to avoid.

`__pos` Index of character to search back from (default end).

### Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1853 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_not_of()`.

```
5.16.3.59 template<typename _CharT, typename _Traits, typename _Alloc
 , template< typename, typename, typename > class _Base>
 __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type
 __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
 >::find_last_of(const _CharT * __s, size_type __pos, size_type
 __n) const
```

Find last position of a character of C substring.

**Parameters**

- `__s` C string containing characters to locate.
- `__pos` Index of character to search back from.
- `__n` Number of characters from `s` to search for.

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 370 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::npos`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**5.16.3.60** `template<typename _CharT, typename _Traits, typename _Alloc,`  
`template< typename, typename, typename > class _Base> size_type`  
`__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base`  
`>::find_last_of ( const __versa_string< _CharT, _Traits, _Alloc,`  
`_Base > & __str, size_type __pos = npos ) const [inline]`

Find last position of a character of string.

**Parameters**

- `__str` String containing characters to locate.
- `__pos` Index of character to search back from (default end).

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1699 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::data()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_of()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_of()`.

**5.16.3.61** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base> size_type  
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base  
>::find_last_of ( const _CharT * __s, size_type __pos = npos )  
const [inline]`

Find last position of a character of C string.

#### Parameters

`__s` C string containing characters to locate.

`__pos` Index of character to search back from (default end).

#### Returns

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1728 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_of()`.

**5.16.3.62** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base> size_type  
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base  
>::find_last_of ( _CharT __c, size_type __pos = npos ) const  
[inline]`

Find last position of a character.

#### Parameters

`__c` Character to locate.

`__pos` Index of character to search back from (default end).

#### Returns

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `rfind(c, pos)`.

Definition at line 1747 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::rfind()`.

**5.16.3.63** `template<typename _CharT, typename _Traits, typename _Alloc,`  
`template< typename, typename, typename > class _Base> reference`  
`__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::front`  
`( ) [inline]`

Returns a read/write reference to the data at the first element of the string.

Definition at line 602 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base`  
`>::operator[]()`.

**5.16.3.64** `template<typename _CharT, typename _Traits, typename _Alloc,`  
`template< typename, typename, typename > class _Base>`  
`const_reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,`  
`_Base >::front ( ) const [inline]`

Returns a read-only (constant) reference to the data at the first element of the string.

Definition at line 610 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base`  
`>::operator[]()`.

**5.16.3.65** `template<typename _CharT, typename _Traits, typename _Alloc,`  
`template< typename, typename, typename > class _Base>`  
`allocator_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,`  
`_Base >::get_allocator ( ) const [inline]`

Return copy of allocator used to construct this string.

Definition at line 1504 of file `vstring.h`.

**5.16.3.66** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,  
_Alloc, _Base >::insert ( size_type __pos, const _CharT * __s,  
size_type __n ) [inline]`

Insert a C substring.

#### Parameters

`__pos` Iterator referencing location in string to insert at.

`__s` The C string to insert.

`__n` The number of characters to insert.

#### Returns

Reference to this string.

#### Exceptions

[\*`std::length\_error`\*](#) If new length exceeds `max_size()`.

[\*`std::out\_of\_range`\*](#) If `__pos` is beyond the end of this string.

Inserts the first `__n` characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1004 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

**5.16.3.67** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base> iterator  
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert  
( iterator __p, _CharT __c ) [inline]`

Insert one character.

#### Parameters

`__p` Iterator referencing position in string to insert at.

`__c` The character to insert.



### Returns

Iterator referencing newly inserted char.

### Exceptions

*[std::length\\_error](#)* If new length exceeds `max_size()`.

Inserts character `__c` at position referenced by `__p`. If adding character causes the length to exceed `max_size()`, `length_error` is thrown. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1065 of file `vstring.h`.

**5.16.3.68** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,  
_Alloc, _Base>::insert ( size_type __pos1, const __versa_string<  
_CharT, _Traits, _Alloc, _Base> & __str ) [inline]`

Insert value of a string.

### Parameters

`__pos1` Iterator referencing location in string to insert at.

`__str` The string to insert.

### Returns

Reference to this string.

### Exceptions

*[std::length\\_error](#)* If new length exceeds `max_size()`.

Inserts value of `__str` starting at `__pos1`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 958 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**5.16.3.69** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base> void  
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert  
( iterator __p, size_type __n, _CharT __c ) [inline]`

Insert multiple characters.

#### Parameters

- `__p` Iterator referencing location in string to insert at.
- `__n` Number of characters to insert
- `__c` The character to insert.

#### Exceptions

[`std::length\_error`](#) If new length exceeds `max_size()`.

Inserts `__n` copies of character `__c` starting at the position referenced by iterator `__p`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 913 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

**5.16.3.70** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base> void  
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert  
( iterator __p, std::initializer_list<_CharT> __l ) [inline]`

Insert an `initializer_list` of characters.

#### Parameters

- `__p` Iterator referencing location in string to insert at.
- `__l` The `initializer_list` of characters to insert.

#### Exceptions

[`std::length\_error`](#) If new length exceeds `max_size()`.

Definition at line 941 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert()`.

**5.16.3.71** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
template<class _InputIterator > void __gnu_cxx::__versa_string<  
_CharT, _Traits, _Alloc, _Base >::insert ( iterator __p,  
_InputIterator __beg, _InputIterator __end ) [inline]`

Insert a range of characters.

#### Parameters

`__p` Iterator referencing location in string to insert at.  
`__beg` Start of range.  
`__end` End of range.

#### Exceptions

`std::length_error` If new length exceeds `max_size()`.

Inserts characters in range `[beg,end)`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 930 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

**5.16.3.72** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits,  
_Alloc, _Base >::insert ( size_type __pos, size_type __n, _CharT  
__c ) [inline]`

Insert multiple characters.

#### Parameters

`__pos` Index in string to insert at.  
`__n` Number of characters to insert  
`__c` The character to insert.

#### Returns

Reference to this string.

### Exceptions

*`std::length_error`* If new length exceeds `max_size()`.

*`std::out_of_range`* If `__pos` is beyond the end of this string.

Inserts `__n` copies of character `__c` starting at index `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos > length()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1047 of file `vstring.h`.

**5.16.3.73** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,  
_Alloc, _Base>::insert ( size_type __pos1, const __versa_string<  
_CharT, _Traits, _Alloc, _Base> & __str, size_type __pos2,  
size_type __n ) [inline]`

Insert a substring.

### Parameters

`__pos1` Iterator referencing location in string to insert at.

`__str` The string to insert.

`__pos2` Start of characters in `str` to insert.

`__n` Number of characters to insert.

### Returns

Reference to this string.

### Exceptions

*`std::length_error`* If new length exceeds `max_size()`.

*`std::out_of_range`* If `__pos1 > size()` or `__pos2 > __str.size()`.

Starting at `__pos1`, insert `__n` character of `__str` beginning with `__pos2`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos1` is beyond the end of this string or `__pos2` is beyond the end of `__str`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 981 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

**5.16.3.74** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,  
_Alloc, _Base >::insert ( size_type __pos, const _CharT * __s )  
[inline]`

Insert a C string.

#### Parameters

`__pos` Iterator referencing location in string to insert at.

`__s` The C string to insert.

#### Returns

Reference to this string.

#### Exceptions

[\*`std::length\_error`\*](#) If new length exceeds `max_size()`.

[\*`std::out\_of\_range`\*](#) If `__pos` is beyond the end of this string.

Inserts the first `__n` characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1023 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

**5.16.3.75** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base> size_type  
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base  
>::length ( ) const [inline]`

Returns the number of characters in the string, not including any /// null-termination.

Definition at line 428 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`.

**5.16.3.76** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base> size_type  
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base  
>::max_size ( ) const [inline]`

Returns the [size\(\)](#) of the largest possible string.

Definition at line 433 of file `vstring.h`.

Referenced by `std::getline()`.

**5.16.3.77** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,  
_Alloc, _Base >::operator+=( const __versa_string< _CharT,  
_Traits, _Alloc, _Base > & __str ) [inline]`

Append a string to this string.

#### Parameters

`__str` The string to append.

#### Returns

Reference to this string.

Definition at line 637 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append()`.

**5.16.3.78** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,  
_Alloc, _Base >::operator+=( const _CharT * __s ) [inline]`

Append a C string.

#### Parameters

`__s` The C string to append.

### Returns

Reference to this string.

Definition at line 646 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append()`.

**5.16.3.79** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits,  
_Alloc, _Base>::operator+=( _CharT __c ) [inline]`

Append a character.

### Parameters

`__c` The character to append.

### Returns

Reference to this string.

Definition at line 655 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::push_back()`.

**5.16.3.80** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits,  
_Alloc, _Base>::operator+=( std::initializer_list<_CharT> __l )  
[inline]`

Append an `initializer_list` of characters.

### Parameters

`__l` The `initializer_list` of characters to be appended.

### Returns

Reference to this string.

Definition at line 668 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append()`.

**5.16.3.81** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,  
_Alloc, _Base >::operator= ( const _CharT * __s ) [inline]`

Copy contents of `__s` into this string.

**Parameters**

`__s` Source null-terminated string.

Definition at line 289 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign()`.

**5.16.3.82** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,  
_Alloc, _Base >::operator= ( std::initializer_list< _CharT > __l )  
[inline]`

Set value to string constructed from initializer list.

**Parameters**

`__l` [std::initializer\\_list](#).

Definition at line 277 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign()`.

**5.16.3.83** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,  
_Alloc, _Base >::operator= ( __versa_string< _CharT, _Traits,  
_Alloc, _Base > && __str ) [inline]`

String move assignment operator.

**Parameters**

`__str` Source string.



The contents of `__str` are moved into this string (without copying). `__str` is a valid, but unspecified string.

Definition at line 265 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::swap()`.

**5.16.3.84** `template<typename _CharT, typename _Traits, typename _Alloc,`  
`template< typename, typename, typename > class _Base>`  
`__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits,`  
`_Alloc, _Base>::operator=( _CharT __c ) [inline]`

Set value to string of length 1.

#### Parameters

`__c` Source character.

Assigning to a character makes this string length 1 and `(*this)[0] == __c`.

Definition at line 300 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::assign()`.

**5.16.3.85** `template<typename _CharT, typename _Traits, typename _Alloc,`  
`template< typename, typename, typename > class _Base>`  
`__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits,`  
`_Alloc, _Base>::operator=( const __versa_string<_CharT, _Traits,`  
`_Alloc, _Base> & __str ) [inline]`

Assign the value of `str` to this string.

#### Parameters

`__str` Source string.

Definition at line 253 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::assign()`.

**5.16.3.86** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
const_reference __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc,  
_Base>::operator[] ( size_type __pos ) const [inline]`

Subscript access to the data contained in the string.

#### Parameters

`__pos` The index of the character to access.

#### Returns

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see [at\(\)](#).)

Definition at line 531 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::back()`,  
and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::front()`.

**5.16.3.87** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base> reference  
__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base  
>::operator[] ( size_type __pos ) [inline]`

Subscript access to the data contained in the string.

#### Parameters

`__pos` The index of the character to access.

#### Returns

Read/write reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see [at\(\)](#).) Unshares the string.

Definition at line 548 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**5.16.3.88** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base> void  
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base  
>::push_back ( _CharT __c ) [inline]`

Append a single character.

#### Parameters

`__c` Character to append.

Definition at line 769 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::assign()`,  
`__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::capacity()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `__gnu_cxx::operator+()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::operator+=()`.

**5.16.3.89** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits,  
_Alloc, _Base>::rbegin ( ) [inline]`

Returns a read/write reverse iterator that points to the last character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 351 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::end()`.

**5.16.3.90** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
const_reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits,  
_Alloc, _Base>::rbegin ( ) const [inline]`

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 360 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::end()`.

**5.16.3.91** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
const_reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits,  
_Alloc, _Base >::rend ( ) const [inline]`

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 378 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::begin()`.

**5.16.3.92** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits,  
_Alloc, _Base >::rend ( ) [inline]`

Returns a read/write reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 369 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::begin()`.

**5.16.3.93** `template<typename _CharT, typename _Traits, typename  
_Alloc, template< typename, typename, typename > class  
_Base> __versa_string& __gnu_cxx::__versa_string< _CharT,  
_Traits, _Alloc, _Base >::replace ( iterator __i1, iterator __i2,  
std::initializer_list<_CharT> __l ) [inline]`

Replace range of characters with `initializer_list`.

#### Parameters

`__i1` Iterator referencing start of range to replace.

`__i2` Iterator referencing end of range to replace.

`__l` The `initializer_list` of characters to insert.

#### Returns

Reference to this string.

#### Exceptions

[\*`std::length\_error`\*](#) If new length exceeds `max_size()`.

Removes the characters in the range [i1,i2). In place, characters in the range [k1,k2) are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1423 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

**5.16.3.94** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace ( iterator __i1, iterator __i2, const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str ) [inline]`

Replace range of characters with string.

#### Parameters

- `__i1` Iterator referencing start of range to replace.
- `__i2` Iterator referencing end of range to replace.
- `__str` String value to insert.

#### Returns

Reference to this string.

#### Exceptions

`std::length_error` If new length exceeds `max_size()`.

Removes the characters in the range [i1,i2). In place, the value of `__str` is inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1272 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

**5.16.3.95** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,  
_Alloc, _Base >::replace ( size_type __pos, size_type __n, const  
__versa_string< _CharT, _Traits, _Alloc, _Base > & __str )  
[inline]`

Replace characters with value from another string.

#### Parameters

`__pos` Index of first character to replace.  
`__n` Number of characters to be replaced.  
`__str` String to insert.

#### Returns

Reference to this string.

#### Exceptions

[\*std::out\\_of\\_range\*](#) If `__pos` is beyond the end of this string.  
[\*std::length\\_error\*](#) If new length exceeds `max_size()`.

Removes the characters in the range `[pos,pos+n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1155 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::append()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::insert()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`.

**5.16.3.96** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,  
_Alloc, _Base >::replace ( size_type __pos, size_type __n1,  
size_type __n2, _CharT __c ) [inline]`

Replace characters with multiple characters.

#### Parameters

- `__pos` Index of first character to replace.
- `__n1` Number of characters to be replaced.
- `__n2` Number of characters to insert.
- `__c` Character to insert.

#### Returns

Reference to this string.

#### Exceptions

- [\*`std::out\_of\_range`\*](#) If `__pos > size()`.
- [\*`std::length\_error`\*](#) If new length exceeds `max_size()`.

Removes the characters in the range `[pos, pos + n1)` from this string. In place, `__n2` copies of `__c` are inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1254 of file `vstring.h`.

**5.16.3.97** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits,  
_Alloc, _Base>::replace ( iterator __i1, iterator __i2, size_type  
__n, _CharT __c ) [inline]`

Replace range of characters with multiple characters.

#### Parameters

- `__i1` Iterator referencing start of range to replace.
- `__i2` Iterator referencing end of range to replace.
- `__n` Number of characters to insert.
- `__c` Character to insert.

#### Returns

Reference to this string.

### Exceptions

*`std::length_error`* If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, `__n` copies of `__c` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1332 of file `vstring.h`.

**5.16.3.98** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,  
_Alloc, _Base >::replace ( iterator __i1, iterator __i2, const  
_CharT * __s ) [inline]`

Replace range of characters with C string.

### Parameters

`__i1` Iterator referencing start of range to replace.

`__i2` Iterator referencing end of range to replace.

`__s` C string value to insert.

### Returns

Reference to this string.

### Exceptions

*`std::length_error`* If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, the characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1311 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.



**5.16.3.99** `template<typename _CharT, typename _Traits, typename  
_Alloc, template< typename, typename, typename > class  
_Base> template<class _InputIterator > __versa_string&  
__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base  
>::replace ( iterator __i1, iterator __i2, _InputIterator __k1,  
_InputIterator __k2 ) [inline]`

Replace range of characters with range.

#### Parameters

- `__i1` Iterator referencing start of range to replace.
- `__i2` Iterator referencing end of range to replace.
- `__k1` Iterator referencing start of range to insert.
- `__k2` Iterator referencing end of range to insert.

#### Returns

Reference to this string.

#### Exceptions

[\*`std::length\_error`\*](#) If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, characters in the range `[k1,k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1355 of file `vstring.h`.

**5.16.3.100** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits,  
_Alloc, _Base>::replace ( size_type __pos, size_type __n1, const  
_CharT * __s, size_type __n2 ) [inline]`

Replace characters with value of a C substring.

#### Parameters

- `__pos` Index of first character to replace.
- `__n1` Number of characters to be replaced.

`__s` C string to insert.

`__n2` Number of characters from `__s` to use.

### Returns

Reference to this string.

### Exceptions

*[std::out\\_of\\_range](#)* If `__pos1 > size()`.

*[std::length\\_error](#)* If new length exceeds `max_size()`.

Removes the characters in the range `[pos, pos + n1)` from this string. In place, the first `__n2` characters of `__s` are inserted, or all of `__s` if `__n2` is too large. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1206 of file `vstring.h`.

**5.16.3.101** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits,  
_Alloc, _Base>::replace ( size_type __pos, size_type __n1, const  
_CharT * __s ) [inline]`

Replace characters with value of a C string.

### Parameters

`__pos` Index of first character to replace.

`__n1` Number of characters to be replaced.

`__s` C string to insert.

### Returns

Reference to this string.

### Exceptions

*[std::out\\_of\\_range](#)* If `__pos > size()`.

*[std::length\\_error](#)* If new length exceeds `max_size()`.

Removes the characters in the range `[pos, pos + n1)` from this string. In place, the characters of `__s` are inserted. If `pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1230 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

**5.16.3.102** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits,  
_Alloc, _Base>::replace ( size_type __pos1, size_type __n1,  
const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str,  
size_type __pos2, size_type __n2 ) [inline]`

Replace characters with value from another string.

#### Parameters

- `__pos1` Index of first character to replace.
- `__n1` Number of characters to be replaced.
- `__str` String to insert.
- `__pos2` Index of first character of `str` to use.
- `__n2` Number of characters from `str` to use.

#### Returns

Reference to this string.

#### Exceptions

- `std::out_of_range` If `__pos1 > size()` or `__pos2 > str.size()`.
- `std::length_error` If new length exceeds `max_size()`.

Removes the characters in the range `[pos1, pos1 + n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1178 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

**5.16.3.103** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,  
_Alloc, _Base >::replace ( iterator __i1, iterator __i2, const  
_CharT * __s, size_type __n ) [inline]`

Replace range of characters with C substring.

#### Parameters

`__i1` Iterator referencing start of range to replace.  
`__i2` Iterator referencing end of range to replace.  
`__s` C string value to insert.  
`__n` Number of characters from `s` to insert.

#### Returns

Reference to this string.

#### Exceptions

[\*std::length\\_error\*](#) If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, the first `n` characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1290 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

**5.16.3.104** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base> void  
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base  
>::reserve ( size_type __res_arg = 0 ) [inline]`

Attempt to preallocate enough memory for specified number of characters.

#### Parameters

`__res_arg` Number of characters required.

#### Exceptions

[\*std::length\\_error\*](#) If `__res_arg` exceeds `max_size()`.

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

Definition at line 501 of file `vstring.h`.

Referenced by `__gnu_cxx::operator+()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::shrink_to_fit()`.

**5.16.3.105** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::resize ( size_type __n, _CharT __c )`

Resizes the string to the specified number of characters.

#### Parameters

- `__n` Number of characters the string should contain.
- `__c` Character to fill any new elements.

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to `__c`.

Definition at line 51 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**5.16.3.106** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::resize ( size_type __n ) [inline]`

Resizes the string to the specified number of characters.

#### Parameters

- `__n` Number of characters the string should contain.

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as char, this means setting them to 0.

Definition at line 460 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::resize()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::resize()`.

**5.16.3.107** `template<typename _CharT, typename _Traits, typename _Alloc, template<typename, typename, typename> class _Base> __versa_string<_CharT, _Traits, _Alloc, _Base>::size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::rfind ( const _CharT * __s, size_type __pos, size_type __n ) const`

Find last position of a C substring.

#### Parameters

- `__s` C string to locate.
- `__pos` Index of character to search back from.
- `__n` Number of characters from `s` to search for.

#### Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 313 of file `vstring.tcc`.

References `std::min()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::npos`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**5.16.3.108** `template<typename _CharT, typename _Traits, typename _Alloc, template<typename, typename, typename> class _Base> __versa_string<_CharT, _Traits, _Alloc, _Base>::size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::rfind ( _CharT __c, size_type __pos = npos ) const`

Find last position of a character.

#### Parameters

- `__c` Character to locate.
- `__pos` Index of character to search back from (default end).

#### Returns

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 335 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::npos`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

**5.16.3.109** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base> size_type  
__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base  
>::rfind ( const _CharT * __s, size_type __pos = npos ) const  
[inline]`

Find last position of a C string.

#### Parameters

- `__s` C string to locate.
- `__pos` Index of character to start search at (default end).

#### Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1606 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::rfind()`.

**5.16.3.110** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base> size_type  
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base  
>::rfind ( const __versa_string< _CharT, _Traits, _Alloc, _Base >  
& __str, size_type __pos = npos ) const [inline]`

Find last position of a string.

#### Parameters

`__str` String to locate.

`__pos` Index of character to search back from (default end).

#### Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1577 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::data()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::rfind()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_of()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::rfind()`.

**5.16.3.111** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base> void  
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base  
>::shrink_to_fit ( ) [inline]`

A non-binding request to reduce `capacity()` to `size()`.

Definition at line 466 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::reserve()`.



**5.16.3.112** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base> size_type  
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size  
( ) const [inline]`

Returns the number of characters in the string, not including any /// null-termination.

Definition at line 422 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::at()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::back()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::back()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::back()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::empty()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::end()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_not_of()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_not_of()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert()`, `__gnu_cxx::operator+()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator[]()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator[]()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::push_back()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::push_back()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::resize()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rfind()`.

**5.16.3.113** `template<typename _CharT, typename _Traits, typename _Alloc,  
template< typename, typename, typename > class _Base>  
__versa_string __gnu_cxx::__versa_string< _CharT, _Traits,  
_Alloc, _Base >::substr ( size_type __pos = 0, size_type __n = npos  
) const [inline]`

Get a substring.

#### Parameters

`__pos` Index of first character (default 0).

`__n` Number of characters in substring (default remainder).

#### Returns

The new string.

## Exceptions

*`std::out_of_range`* If `pos > size()`.

Construct and return a new string using the `__n` characters starting at `__pos`. If the string is too short, use the remainder of the characters. If `__pos` is beyond the end of the string, `out_of_range` is thrown.

Definition at line 1885 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::__versa_string()`.

**5.16.3.114** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::swap ( __versa_string<_CharT, _Traits, _Alloc, _Base> & __s ) [inline]`

Swap contents with another string.

## Parameters

`__s` String to swap with.

Exchanges the contents of this string with that of `__s` in constant time.

Definition at line 1476 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::assign()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::operator=()`, and `__gnu_cxx::swap()`.

## 5.16.4 Member Data Documentation

**5.16.4.1** `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const __versa_string<_CharT, _Traits, _Alloc, _Base>::size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::npos [static]`

Value returned by various member functions when they fail.

Definition at line 79 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_not_of()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_of()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_not_of()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_of()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::rfind()`.

The documentation for this class was generated from the following files:

- [vstring.h](#)
- [vstring.tcc](#)

## 5.17 `__gnu_cxx::_Caster<_ToType>` Struct Template Reference

### Public Types

- `typedef _ToType::element_type * type`

#### 5.17.1 Detailed Description

`template<typename _ToType> struct __gnu_cxx::_Caster<_ToType>`

These functions are here to allow containers to support non standard pointer types. For normal pointers, these resolve to the use of the standard cast operation. For other types the functions will perform the appropriate cast to/from the custom pointer class so long as that class meets the following conditions: 1) has a `typedef element_type` which names the type it points to. 2) has a `get()` const method which returns `element_type*`. 3) has a constructor which can take one `element_type*` argument. This type supports the semantics of the pointer cast operators (below.)

Definition at line 52 of file `cast.h`.

The documentation for this struct was generated from the following file:

- [cast.h](#)

## 5.18 `__gnu_cxx::_Char_types<_CharT>` Struct Template Reference

Mapping from character type to associated types.

### Public Types

- `typedef unsigned long int_type`

- typedef `std::streamoff` **off\_type**
- typedef `std::streampos` **pos\_type**
- typedef `std::mbstate_t` **state\_type**

### 5.18.1 Detailed Description

**template<typename \_CharT> struct `__gnu_cxx::_Char_types<_CharT>`**

Mapping from character type to associated types.

#### Note

This is an implementation class for the generic version of `char_traits`. It defines `int_type`, `off_type`, `pos_type`, and `state_type`. By default these are unsigned long, `streamoff`, `streampos`, and `mbstate_t`. Users who need a different set of types, but who don't need to change the definitions of any function defined in `char_traits`, can specialize `__gnu_cxx::_Char_types` while leaving `__gnu_cxx::char_traits` alone.

Definition at line 60 of file `char_traits.h`.

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)

## 5.19 `__gnu_cxx::_ExtPtr_allocator<_Tp>` Class Template Reference

An example allocator which uses a non-standard pointer type.

This allocator specifies that containers use a 'relative pointer' as it's pointer type. (See [ext/pointer.h](#)) Memory allocation in this example is still performed using `std::allocator`.

### Public Types

- typedef `_Pointer_adapter<_Relative_pointer_impl<const _Tp>>` **const\_pointer**
- typedef `const _Tp &` **const\_reference**
- typedef `std::ptrdiff_t` **difference\_type**
- typedef `_Pointer_adapter<_Relative_pointer_impl<_Tp>>` **pointer**
- typedef `_Tp &` **reference**
- typedef `std::size_t` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- `_ExtPtr_allocator` (const `_ExtPtr_allocator` &\_\_rarg) throw ()
- template<typename `_Up` >  
`_ExtPtr_allocator` (const `_ExtPtr_allocator`< `_Up` > &\_\_rarg) throw ()
- const `std::allocator`< `_Tp` > & `_M_getUnderlyingImp` () const
- `pointer address` (reference \_\_x) const
- `const_pointer address` (const\_reference \_\_x) const
- `pointer allocate` (size\_type \_\_n, void \*\_\_hint=0)
- void `construct` (`pointer` \_\_p, const `_Tp` &\_\_val)
- template<typename... `_Args`>  
void `construct` (`pointer` \_\_p, `_Args` &&...\_\_args)
- void `deallocate` (`pointer` \_\_p, size\_type \_\_n)
- void `destroy` (`pointer` \_\_p)
- size\_type `max_size` () const throw ()
- template<typename `_Up` >  
bool `operator!=` (const `_ExtPtr_allocator`< `_Up` > &\_\_rarg)
- bool `operator!=` (const `_ExtPtr_allocator` &\_\_rarg)
- template<typename `_Up` >  
bool `operator==` (const `_ExtPtr_allocator`< `_Up` > &\_\_rarg)
- bool `operator==` (const `_ExtPtr_allocator` &\_\_rarg)

## Friends

- template<typename `_Up` >  
void `swap` (`_ExtPtr_allocator`< `_Up` > &, `_ExtPtr_allocator`< `_Up` > &)

### 5.19.1 Detailed Description

template<typename `_Tp`> class `__gnu_cxx::__ExtPtr_allocator`< `_Tp` >

An example allocator which uses a non-standard pointer type.

This allocator specifies that containers use a 'relative pointer' as it's pointer type. (See [ext/pointer.h](#)) Memory allocation in this example is still performed using `std::allocator`.

Definition at line 56 of file `extptr_allocator.h`.

The documentation for this class was generated from the following file:

- [extptr\\_allocator.h](#)

## 5.20 `__gnu_cxx::_Invalid_type` Struct Reference

### 5.20.1 Detailed Description

The specialization on this type helps resolve the problem of reference to void, and eliminates the need to specialize `_Pointer_adapter` for cases of void\*, const void\*, and so on.

Definition at line 209 of file `pointer.h`.

The documentation for this struct was generated from the following file:

- [pointer.h](#)

## 5.21 `__gnu_cxx::_Pointer_adapter< _Storage_policy >` Class Template Reference

Inherits `_Storage_policy`.

### Public Types

- typedef `std::ptrdiff_t` **difference\_type**
- typedef `_Storage_policy::element_type` **element\_type**
- typedef `std::random_access_iterator_tag` **iterator\_category**
- typedef `_Pointer_adapter` **pointer**
- typedef `_Reference_type< element_type >::reference` **reference**
- typedef `_Unqualified_type< element_type >::type` **value\_type**

### Public Member Functions

- `_Pointer_adapter` (`element_type *` \_\_arg=0)
- `_Pointer_adapter` (const [\\_Pointer\\_adapter](#) &\_\_arg)
- template<typename `_Up` >  
  `_Pointer_adapter` (const [\\_Pointer\\_adapter](#)< `_Up` > &\_\_arg)
- template<typename `_Up` >  
  `_Pointer_adapter` (`_Up *` \_\_arg)
- `operator __unspecified_bool_type` () const
- `bool operator!` () const
- reference `operator*` () const
- [\\_Pointer\\_adapter](#) & `operator++` ()
- [\\_Pointer\\_adapter](#) `operator++` (int)
- [\\_Pointer\\_adapter](#) & `operator+=` (unsigned short \_\_offset)
- [\\_Pointer\\_adapter](#) & `operator+=` (int \_\_offset)

- `_Pointer_adapter` & **operator+=** (unsigned int \_\_offset)
- `_Pointer_adapter` & **operator+=** (long \_\_offset)
- `_Pointer_adapter` & **operator+=** (unsigned long \_\_offset)
- `_Pointer_adapter` & **operator+=** (short \_\_offset)
- `template<typename _Up >`  
`std::ptrdiff_t operator- (const _Pointer_adapter<_Up> &__rhs) const`
- `_Pointer_adapter` **operator--** (int)
- `_Pointer_adapter` & **operator--** ()
- `_Pointer_adapter` & **operator-=** (unsigned short \_\_offset)
- `_Pointer_adapter` & **operator-=** (short \_\_offset)
- `_Pointer_adapter` & **operator-=** (int \_\_offset)
- `_Pointer_adapter` & **operator-=** (long \_\_offset)
- `_Pointer_adapter` & **operator-=** (unsigned long \_\_offset)
- `_Pointer_adapter` & **operator-=** (unsigned int \_\_offset)
- `element_type * operator-> () const`
- `_Pointer_adapter` & **operator=** (const `_Pointer_adapter` &\_\_arg)
- `template<typename _Up >`  
`_Pointer_adapter & operator= (_Up *__arg)`
- `template<typename _Up >`  
`_Pointer_adapter & operator= (const _Pointer_adapter<_Up> &__arg)`
- reference `operator[] (std::ptrdiff_t __index) const`

#### Friends

- `_Pointer_adapter` **operator+** (const `_Pointer_adapter` &\_\_lhs, short \_\_offset)
- `_Pointer_adapter` **operator+** (const `_Pointer_adapter` &\_\_lhs, unsigned short \_\_offset)
- `_Pointer_adapter` **operator+** (unsigned long \_\_offset, const `_Pointer_adapter` &\_\_rhs)
- `_Pointer_adapter` **operator+** (unsigned short \_\_offset, const `_Pointer_adapter` &\_\_rhs)
- `_Pointer_adapter` **operator+** (const `_Pointer_adapter` &\_\_lhs, long \_\_offset)
- `_Pointer_adapter` **operator+** (long \_\_offset, const `_Pointer_adapter` &\_\_rhs)
- `_Pointer_adapter` **operator+** (const `_Pointer_adapter` &\_\_lhs, int \_\_offset)
- `_Pointer_adapter` **operator+** (const `_Pointer_adapter` &\_\_lhs, unsigned long \_\_offset)
- `_Pointer_adapter` **operator+** (unsigned int \_\_offset, const `_Pointer_adapter` &\_\_rhs)
- `_Pointer_adapter` **operator+** (short \_\_offset, const `_Pointer_adapter` &\_\_rhs)
- `_Pointer_adapter` **operator+** (int \_\_offset, const `_Pointer_adapter` &\_\_rhs)
- `_Pointer_adapter` **operator+** (const `_Pointer_adapter` &\_\_lhs, unsigned int \_\_offset)
- `std::ptrdiff_t operator- (element_type *__lhs, const _Pointer_adapter &__rhs)`

- `std::ptrdiff_t operator-` (const [\\_Pointer\\_adapter](#) &\_\_lhs, element\_type \*\_\_rhs)
- [\\_Pointer\\_adapter operator-](#) (const [\\_Pointer\\_adapter](#) &\_\_lhs, unsigned short \_\_offset)
- [\\_Pointer\\_adapter operator-](#) (const [\\_Pointer\\_adapter](#) &\_\_lhs, unsigned long \_\_offset)
- [\\_Pointer\\_adapter operator-](#) (const [\\_Pointer\\_adapter](#) &\_\_lhs, int \_\_offset)
- [\\_Pointer\\_adapter operator-](#) (const [\\_Pointer\\_adapter](#) &\_\_lhs, short \_\_offset)
- `template<typename _Up >`  
`std::ptrdiff_t operator-` (\_Up \*\_\_lhs, const [\\_Pointer\\_adapter](#) &\_\_rhs)
- [\\_Pointer\\_adapter operator-](#) (const [\\_Pointer\\_adapter](#) &\_\_lhs, long \_\_offset)
- `template<typename _Up >`  
`std::ptrdiff_t operator-` (const [\\_Pointer\\_adapter](#) &\_\_lhs, \_Up \*\_\_rhs)
- [\\_Pointer\\_adapter operator-](#) (const [\\_Pointer\\_adapter](#) &\_\_lhs, unsigned int \_\_offset)

### 5.21.1 Detailed Description

`template<typename _Storage_policy> class __gnu_cxx::_Pointer_adapter<_Storage_policy>`

The following provides an 'alternative pointer' that works with the containers when specified as the pointer typedef of the allocator.

The pointer type used with the containers doesn't have to be this class, but it must support the implicit conversions, pointer arithmetic, comparison operators, etc. that are supported by this class, and avoid raising compile-time ambiguities. Because creating a working pointer can be challenging, this pointer template was designed to wrapper an easier storage policy type, so that it becomes reusable for creating other pointer types.

A key point of this class is also that it allows container writers to 'assume' `Allocator::pointer` is a typedef for a normal pointer. This class supports most of the conventions of a true pointer, and can, for instance handle implicit conversion to const and base class pointer types. The only impositions on container writers to support extended pointers are: 1) use the `Allocator::pointer` typedef appropriately for pointer types. 2) if you need pointer casting, use the `__pointer_cast<>` functions from [ext/cast.h](#). This allows pointer cast operations to be overloaded is necessary by custom pointers.

Note: The const qualifier works with this pointer adapter as follows:

```
_Tp* == _Pointer_adapter<_Std_pointer_impl<_Tp>> >; const _Tp* == _Pointer_adapter<_Std_pointer_impl<const _Tp>> >;
_Tp* const == const _Pointer_adapter<_Std_pointer_impl<_Tp>> >; const _Tp* const == const _Pointer_adapter<_Std_pointer_impl<const _Tp>> >;
```

Definition at line 285 of file `pointer.h`.

The documentation for this class was generated from the following file:



## 5.22 `__gnu_cxx::_Relative_pointer_impl<_Tp>` Class Template Reference<sup>910</sup>

---

- [pointer.h](#)

## 5.22 `__gnu_cxx::_Relative_pointer_impl<_Tp>` Class Template Reference

A storage policy for use with `_Pointer_adapter<>` which stores the pointer's address as an offset value which is relative to its own address.

### Public Types

- `typedef _Tp element_type`

### Public Member Functions

- `_Tp * get () const`
- `bool operator< (const \_Relative\_pointer\_impl &__rarg) const`
- `bool operator== (const \_Relative\_pointer\_impl &__rarg) const`
- `void set (_Tp *__arg)`

### 5.22.1 Detailed Description

**template<typename \_Tp> class `__gnu_cxx::_Relative_pointer_impl<_Tp>`**

A storage policy for use with `_Pointer_adapter<>` which stores the pointer's address as an offset value which is relative to its own address. This is intended for pointers within shared memory regions which might be mapped at different addresses by different processes. For null pointers, a value of 1 is used. (0 is legitimate sometimes for nodes in circularly linked lists) This value was chosen as the least likely to generate an incorrect null. As there is no reason why any normal pointer would point 1 byte into its own pointer address.

Definition at line 105 of file `pointer.h`.

The documentation for this class was generated from the following file:

- [pointer.h](#)

## 5.23 `__gnu_cxx::_Relative_pointer_impl< const _Tp >` Class Template Reference

### Public Types

- `typedef const _Tp element_type`

**Public Member Functions**

- `const _Tp * get () const`
- `bool operator< (const \_Relative\_pointer\_impl &__rarg) const`
- `bool operator== (const \_Relative\_pointer\_impl &__rarg) const`
- `void set (const _Tp *__arg)`

**5.23.1 Detailed Description**

`template<typename _Tp> class __gnu_cxx::Relative_pointer_impl< const _Tp >`

`Relative_pointer_impl` needs a specialization for `const T` because of the casting done during pointer arithmetic.

Definition at line 157 of file `pointer.h`.

The documentation for this class was generated from the following file:

- [pointer.h](#)

**5.24 `__gnu_cxx::Std_pointer_impl<_Tp>` Class Template Reference**

A storage policy for use with `_Pointer_adapter<>` which yields a standard pointer.

**Public Types**

- `typedef _Tp element_type`

**Public Member Functions**

- `_Tp * get () const`
- `bool operator< (const \_Std\_pointer\_impl &__rarg) const`
- `bool operator== (const \_Std\_pointer\_impl &__rarg) const`
- `void set (element_type *__arg)`

**5.24.1 Detailed Description**

`template<typename _Tp> class __gnu_cxx::Std_pointer_impl< _Tp >`

A storage policy for use with `_Pointer_adapter<>` which yields a standard pointer. A `_Storage_policy` is required to provide 4 things: 1) A `get()` API for returning the stored

pointer value. 2) An `set()` API for storing a pointer value. 3) An `element_type` typedef to define the type this points to. 4) An `operator<()` to support pointer comparison. 5) An `operator==(())` to support pointer comparison.

Definition at line 62 of file `pointer.h`.

The documentation for this class was generated from the following file:

- [pointer.h](#)

## 5.25 `__gnu_cxx::_Unqualified_type<_Tp>` Struct Template Reference

### Public Types

- typedef `_Tp type`

#### 5.25.1 Detailed Description

`template<typename _Tp> struct __gnu_cxx::_Unqualified_type<_Tp>`

This structure accomodates the way in which `std::iterator_traits<>` is normally specialized for `const T*`, so that `value_type` is still `T`.

Definition at line 237 of file `pointer.h`.

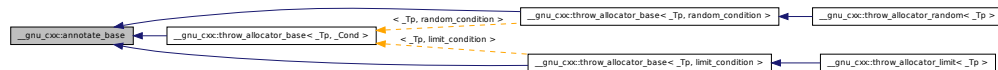
The documentation for this struct was generated from the following file:

- [pointer.h](#)

## 5.26 `__gnu_cxx::annotate_base` Struct Reference

Base class for checking address and label information about allocations. Create a [std::map](#) between the allocated address (`void*`) and a datum for annotations, which are a pair of numbers corresponding to label and allocated size.

Inheritance diagram for `__gnu_cxx::annotate_base`:



## 5.27 `__gnu_cxx::array_allocator<_Tp, _Array>` Class Template Reference 913

### Public Member Functions

- void **check\_allocated** (size\_t label)
- void **check\_allocated** (void \*p, size\_t size)
- void **erase** (void \*p, size\_t size)
- void **insert** (void \*p, size\_t size)

### Static Public Member Functions

- static size\_t **get\_label** ()
- static void **set\_label** (size\_t l)

### Friends

- [std::ostream](#) & **operator<<** ([std::ostream](#) &, const [annotate\\_base](#) &)

#### 5.26.1 Detailed Description

Base class for checking address and label information about allocations. Create a [std::map](#) between the allocated address (void\*) and a datum for annotations, which are a pair of numbers corresponding to label and allocated size.

Definition at line 96 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

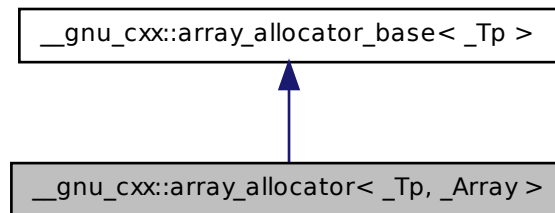
- [throw\\_allocator.h](#)

## 5.27 `__gnu_cxx::array_allocator<_Tp, _Array>` Class Template Reference

An allocator that uses previously allocated memory. This memory can be externally, globally, or otherwise allocated.

## 5.27 `__gnu_cxx::array_allocator<_Tp, _Array>` Class Template Reference 914

Inheritance diagram for `__gnu_cxx::array_allocator<_Tp, _Array>`:



### Public Types

- typedef `_Array` **array\_type**
- typedef `const _Tp *` **const\_pointer**
- typedef `const _Tp &` **const\_reference**
- typedef `ptrdiff_t` **difference\_type**
- typedef `_Tp *` **pointer**
- typedef `_Tp &` **reference**
- typedef `size_t` **size\_type**
- typedef `_Tp` **value\_type**

### Public Member Functions

- **array\_allocator** (`array_type *__array=0`) `throw ()`
- **array\_allocator** (`const array\_allocator &__o`) `throw ()`
- `template<typename _Tp1, typename _Array1 >`  
  **array\_allocator** (`const array\_allocator<_Tp1, _Array1> &`) `throw ()`
- `pointer` **address** (`reference __x`) `const`
- `const_pointer` **address** (`const_reference __x`) `const`
- `pointer` **allocate** (`size_type __n, const void *==0`)
- `void` **construct** (`pointer __p, const _Tp &__val`)
- `template<typename... _Args>`  
  `void` **construct** (`pointer __p, _Args &&...__args`)
- `void` **deallocate** (`pointer, size_type`)
- `void` **destroy** (`pointer __p`)
- `size_type` **max\_size** () `const throw ()`

## 5.27.1 Detailed Description

`template<typename _Tp, typename _Array = std::tr1::array<_Tp, 1>> class _  
__gnu_cxx::array_allocator<_Tp, _Array>`

An allocator that uses previously allocated memory. This memory can be externally, globally, or otherwise allocated.

Definition at line 98 of file `array_allocator.h`.

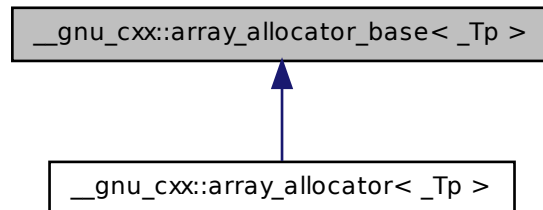
The documentation for this class was generated from the following file:

- [array\\_allocator.h](#)

5.28 `__gnu_cxx::array_allocator_base<_Tp>` Class Template Reference

Base class.

Inheritance diagram for `__gnu_cxx::array_allocator_base<_Tp>`:



## Public Types

- `typedef const _Tp * const_pointer`
- `typedef const _Tp & const_reference`
- `typedef ptrdiff_t difference_type`
- `typedef _Tp * pointer`
- `typedef _Tp & reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

## Public Member Functions

- pointer **address** (reference \_\_x) const
- const\_pointer **address** (const\_reference \_\_x) const
- void **construct** (pointer \_\_p, const \_Tp &\_\_val)
- template<typename... \_Args>  
void **construct** (pointer \_\_p, \_Args &&...\_\_args)
- void **deallocate** (pointer, size\_type)
- void **destroy** (pointer \_\_p)
- size\_type **max\_size** () const throw ()

### 5.28.1 Detailed Description

template<typename \_Tp> class `__gnu_cxx::array_allocator_base< _Tp >`

Base class.

Definition at line 48 of file `array_allocator.h`.

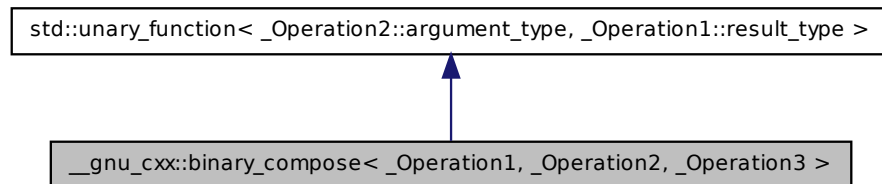
The documentation for this class was generated from the following file:

- [array\\_allocator.h](#)

## 5.29 `__gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >` Class Template Reference

An [SGI extension](#) .

Inheritance diagram for `__gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >`:



### Public Types

- typedef `_Arg` [argument\\_type](#)
- typedef `_Result` [result\\_type](#)

### Public Member Functions

- **binary\_compose** (const `_Operation1` &\_\_x, const `_Operation2` &\_\_y, const `_Operation3` &\_\_z)
- `_Operation1::result_type` **operator()** (const typename `_Operation2::argument_type` &\_\_x) const

### Protected Attributes

- `_Operation1` **\_M\_fn1**
- `_Operation2` **\_M\_fn2**
- `_Operation3` **\_M\_fn3**

#### 5.29.1 Detailed Description

`template<class _Operation1, class _Operation2, class _Operation3> class __gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >`

An [SGI extension](#) .

Definition at line 151 of file `ext/functional`.

#### 5.29.2 Member Typedef Documentation

**5.29.2.1** `template<typename _Arg, typename _Result> typedef _Arg  
std::unary_function< _Arg, _Result >::argument_type  
[inherited]`

`argument_type` is the type of the argument

Definition at line 105 of file `stl_function.h`.

**5.29.2.2** `template<typename _Arg, typename _Result> typedef _Result  
std::unary_function< _Arg, _Result >::result_type [inherited]`



`result_type` is the return type

Definition at line 108 of file `stl_function.h`.

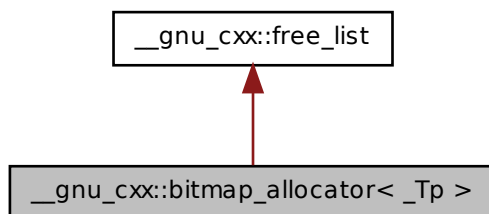
The documentation for this class was generated from the following file:

- [ext/functional](#)

### 5.30 `__gnu_cxx::bitmap_allocator<_Tp>` Class Template Reference

Bitmap Allocator, primary template.

Inheritance diagram for `__gnu_cxx::bitmap_allocator<_Tp>`:



#### Public Types

- `typedef free_list::__mutex_type __mutex_type`
- `typedef const _Tp * const_pointer`
- `typedef const _Tp & const_reference`
- `typedef ptrdiff_t difference_type`
- `typedef _Tp * pointer`
- `typedef _Tp & reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

#### Public Member Functions

- `bitmap_allocator` (const [bitmap\\_allocator](#) &)

- `template<typename _Tp1 >`  
**bitmap\_allocator** (const `bitmap_allocator<_Tp1>` &) throw ()
- pointer `_M_allocate_single_object` () throw (std::bad\_alloc)
- void `_M_deallocate_single_object` (pointer \_\_p) throw ()
- const\_pointer **address** (const\_reference \_\_r) const
- pointer **address** (reference \_\_r) const
- pointer **allocate** (size\_type \_\_n, typename `bitmap_allocator< void >::const_pointer`)
- pointer **allocate** (size\_type \_\_n)
- `template<typename... _Args>`  
void **construct** (pointer \_\_p, \_Args &&...\_\_args)
- void **construct** (pointer \_\_p, const\_reference \_\_data)
- void **deallocate** (pointer \_\_p, size\_type \_\_n) throw ()
- void **destroy** (pointer \_\_p)
- size\_type **max\_size** () const throw ()

### Private Types

- typedef vector\_type::iterator **iterator**
- typedef `__detail::__mini_vector`< value\_type > **vector\_type**

### Private Member Functions

- void `_M_clear` ()
- size\_t \* `_M_get` (size\_t \_\_sz) throw (std::bad\_alloc)
- void `_M_insert` (size\_t \*\_\_addr) throw ()

#### 5.30.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::bitmap_allocator<_Tp>`

Bitmap Allocator, primary template.

Definition at line 691 of file `bitmap_allocator.h`.

#### 5.30.2 Member Function Documentation

**5.30.2.1** `template<typename _Tp> pointer __gnu_cxx::bitmap_allocator<_Tp>::_M_allocate_single_object ( ) throw (std::bad_alloc)`  
**[inline]**

Allocates memory for a single object of size `sizeof(_Tp)`.

### Exceptions

*std::bad\_alloc*. If memory can not be allocated.

Complexity: Worst case complexity is  $O(N)$ , but that is hardly ever hit. If and when this particular case is encountered, the next few cases are guaranteed to have a worst case complexity of  $O(1)$ ! That's why this function performs very well on average. You can consider this function to have a complexity referred to commonly as: Amortized Constant time.

Definition at line 825 of file `bitmap_allocator.h`.

References `__gnu_cxx::__detail::__bit_allocate()`, `__gnu_cxx::__detail::__num_bitmaps()`, and `__gnu_cxx::_Bit_scan_forward()`.

#### 5.30.2.2 `template<typename _Tp> void __gnu_cxx::bitmap_allocator<_Tp>::__M_deallocate_single_object ( pointer __p ) throw () [inline]`

Deallocates memory that belongs to a single object of size `sizeof(_Tp)`.

Complexity:  $O(\lg(N))$ , but the worst case is not hit often! This is because containers usually deallocate memory close to each other and this case is handled in  $O(1)$  time by the `deallocate` function.

Definition at line 915 of file `bitmap_allocator.h`.

References `__gnu_cxx::__detail::__bit_free()`, `__gnu_cxx::__detail::__num_bitmaps()`, `std::__rotate()`, and `__gnu_cxx::free_list::_M_insert()`.

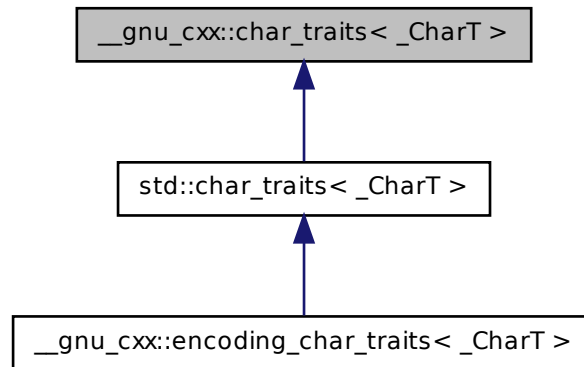
The documentation for this class was generated from the following file:

- [bitmap\\_allocator.h](#)

### 5.31 `__gnu_cxx::char_traits<_CharT>` Struct Template Reference

Base class used to implement [std::char\\_traits](#).

Inheritance diagram for `__gnu_cxx::char_traits<_CharT>`:



### Public Types

- typedef `_CharT` **char\_type**
- typedef `_Char_types<_CharT>::int_type` **int\_type**
- typedef `_Char_types<_CharT>::off_type` **off\_type**
- typedef `_Char_types<_CharT>::pos_type` **pos\_type**
- typedef `_Char_types<_CharT>::state_type` **state\_type**

### Static Public Member Functions

- static void **assign** (`char_type &__c1`, `const char_type &__c2`)
- static `char_type *` **assign** (`char_type *__s`, `std::size_t __n`, `char_type __a`)
- static int **compare** (`const char_type *__s1`, `const char_type *__s2`, `std::size_t __n`)
- static `char_type *` **copy** (`char_type *__s1`, `const char_type *__s2`, `std::size_t __n`)
- static constexpr `int_type` **eof** ()
- static constexpr bool **eq** (`const char_type &__c1`, `const char_type &__c2`)
- static constexpr bool **eq\_int\_type** (`const int_type &__c1`, `const int_type &__c2`)
- static `const char_type *` **find** (`const char_type *__s`, `std::size_t __n`, `const char_type &__a`)

- static `std::size_t length` (`const char_type *__s`)
- static constexpr `bool lt` (`const char_type &__c1, const char_type &__c2`)
- static `char_type * move` (`char_type *__s1, const char_type *__s2, std::size_t __n`)
- static constexpr `int_type not_eof` (`const int_type &__c`)
- static constexpr `char_type to_char_type` (`const int_type &__c`)
- static constexpr `int_type to_int_type` (`const char_type &__c`)

### 5.31.1 Detailed Description

`template<typename _CharT> struct __gnu_cxx::char_traits< _CharT >`

Base class used to implement [std::char\\_traits](#).

#### Note

For any given actual character type, this definition is probably wrong. (Most of the member functions are likely to be right, but the `int_type` and `state_type` typedefs, and the `eof()` member function, are likely to be wrong.) The reason this class exists is so users can specialize it. Classes in namespace `std` may not be specialized for fundamental types, but classes in namespace `__gnu_cxx` may be.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt05ch13s03.html> for advice on how to make use of this class for *unusual* character types. Also, check out [include/ext/pod\\_char\\_traits.h](#).

Definition at line 85 of file `char_traits.h`.

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)

## 5.32 `__gnu_cxx::character< V, I, S >` Struct Template Reference

A POD class that serves as a character abstraction class.

### Public Types

- typedef `character< V, I, S > char_type`
- typedef `I int_type`
- typedef `S state_type`
- typedef `V value_type`

### Static Public Member Functions

- `template<typename V2 >`  
static `char_type` `from` (`const V2 &v`)
- `template<typename V2 >`  
static `V2` `to` (`const char_type &c`)

### Public Attributes

- `value_type` `value`

#### 5.32.1 Detailed Description

`template<typename V, typename I, typename S = std::mbstate_t> struct __gnu_cxx::character< V, I, S >`

A POD class that serves as a character abstraction class.

Definition at line 47 of file `pod_char_traits.h`.

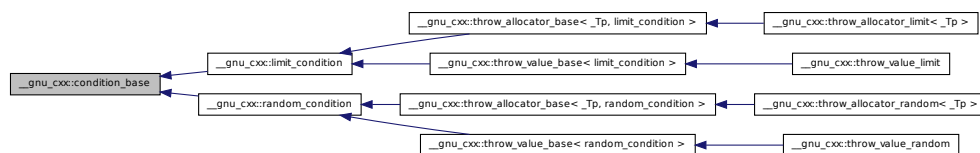
The documentation for this struct was generated from the following file:

- [pod\\_char\\_traits.h](#)

## 5.33 `__gnu_cxx::condition_base` Struct Reference

Base struct for condition policy.

Inheritance diagram for `__gnu_cxx::condition_base`:



#### 5.33.1 Detailed Description

Base struct for condition policy. Requires a public member function with the signature `void throw_conditionally()`

Definition at line 255 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

### 5.34 `__gnu_cxx::constant_binary_fun<_Result, _Arg1, _Arg2>` Struct Template Reference

An [SGI extension](#) .

Inherits `__gnu_cxx::Constant_binary_fun<_Result, _Arg1, _Arg2>`.

#### Public Types

- `typedef _Arg1 first_argument_type`
- `typedef _Result result_type`
- `typedef _Arg2 second_argument_type`

#### Public Member Functions

- `constant_binary_fun` (`const _Result &__v`)
- `const result_type & operator()` (`const _Arg1 &`, `const _Arg2 &`) `const`

#### Public Attributes

- `_Result _M_val`

#### 5.34.1 Detailed Description

`template<class _Result, class _Arg1 = _Result, class _Arg2 = _Arg1> struct __gnu_cxx::constant_binary_fun<_Result, _Arg1, _Arg2>`

An [SGI extension](#) .

Definition at line 317 of file `ext/functional`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

### 5.35 `__gnu_cxx::constant_unary_fun<_Result, _Argument>` Struct Template Reference

An [SGI extension](#) .

## 5.36 `__gnu_cxx::constant_void_fun<_Result>` Struct Template Reference 925

Inherits `__gnu_cxx::_Constant_unary_fun<_Result, _Argument>`.

### Public Types

- typedef `_Argument` **argument\_type**
- typedef `_Result` **result\_type**

### Public Member Functions

- **constant\_unary\_fun** (const `_Result` &\_\_v)
- const `result_type` & **operator()** (const `_Argument` &) const

### Public Attributes

- `result_type` **\_M\_val**

#### 5.35.1 Detailed Description

`template<class _Result, class _Argument = _Result> struct __gnu_cxx::constant_unary_fun<_Result, _Argument>`

An [SGI extension](#).

Definition at line 309 of file `ext/functional`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

## 5.36 `__gnu_cxx::constant_void_fun<_Result>` Struct Template Reference

An [SGI extension](#).

Inherits `__gnu_cxx::_Constant_void_fun<_Result>`.

### Public Types

- typedef `_Result` **result\_type**



### Public Member Functions

- **constant\_void\_fun** (const \_Result &\_\_v)
- const result\_type & **operator()** () const

### Public Attributes

- result\_type **\_M\_val**

#### 5.36.1 Detailed Description

`template<class _Result> struct __gnu_cxx::constant_void_fun<_Result>`

An [SGI extension](#) .

Definition at line 300 of file `ext/functional`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

## 5.37 `__gnu_cxx::debug_allocator<_Alloc>` Class Template Reference

A meta-allocator with debugging bits, as per [20.4].

This is precisely the allocator defined in the C++ Standard.

- all allocation calls `operator new`
- all deallocation calls `operator delete`.

### Public Types

- `typedef _Alloc::const_pointer` **const\_pointer**
- `typedef _Alloc::const_reference` **const\_reference**
- `typedef _Alloc::difference_type` **difference\_type**
- `typedef _Alloc::pointer` **pointer**
- `typedef _Alloc::reference` **reference**
- `typedef _Alloc::size_type` **size\_type**
- `typedef _Alloc::value_type` **value\_type**

## Public Member Functions

- pointer **allocate** (size\_type \_\_n)
- pointer **allocate** (size\_type \_\_n, const void \*\_\_hint)
- void **deallocate** (pointer \_\_p, size\_type \_\_n)

### 5.37.1 Detailed Description

`template<typename _Alloc> class __gnu_cxx::debug_allocator<_Alloc>`

A meta-allocator with debugging bits, as per [20.4].

This is precisely the allocator defined in the C++ Standard.

- all allocation calls operator new
- all deallocation calls operator delete.

Definition at line 63 of file `debug_allocator.h`.

The documentation for this class was generated from the following file:

- [debug\\_allocator.h](#)

## 5.38 `__gnu_cxx::enc_filebuf<_CharT>` Class Template Reference

class [enc\\_filebuf](#).

Inheritance diagram for `__gnu_cxx::enc_filebuf<_CharT>`:



## Public Types

- typedef `codecvt<char_type, char, __state_type>` `__codecvt_type`
- typedef `__basic_file<char>` `__file_type`
- typedef `basic_filebuf<char_type, traits_type>` `__filebuf_type`
- typedef `traits_type::state_type` `__state_type`
- typedef `basic_streambuf<char_type, traits_type>` `__streambuf_type`
- typedef `_CharT` `char_type`

- `typedef traits_type::int_type int_type`
- `typedef traits_type::off_type off_type`
- `typedef traits_type::pos_type pos_type`
- `typedef traits_type::state_type state_type`
- `typedef encoding_char_traits<_CharT> traits_type`

### Public Member Functions

- `enc_filebuf (state_type &__state)`
- `__filebuf_type * close ()`
- `streamsize in_avail ()`
- `bool is_open () const throw ()`
- `__filebuf_type * open (const std::string &__s, ios_base::openmode __mode)`
- `__filebuf_type * open (const char *__s, ios_base::openmode __mode)`
- `int_type sbumpc ()`
- `int_type sgetc ()`
- `streamsize sgetn (char_type *__s, streamsize __n)`
- `int_type snextc ()`
- `int_type sputbackc (char_type __c)`
- `int_type sputc (char_type __c)`
- `streamsize sputn (const char_type *__s, streamsize __n)`
- `int_type sungetc ()`

### Protected Member Functions

- `void _M_allocate_internal_buffer ()`
- `bool _M_convert_to_external (char_type *, streamsize)`
- `void _M_create_pback ()`
- `void _M_destroy_internal_buffer () throw ()`
- `void _M_destroy_pback () throw ()`
- `int _M_get_ext_pos (__state_type &__state)`
- `pos_type _M_seek (off_type __off, ios_base::seekdir __way, __state_type __state)`
- `void _M_set_buffer (streamsize __off)`
- `bool _M_terminate_output ()`
- `void gbump (int __n)`
- `virtual void imbue (const locale &__loc)`
- `virtual int_type overflow (int_type __c=encoding_char_traits<_CharT>::eof())`
- `virtual int_type pbackfail (int_type __c=encoding_char_traits<_CharT>::eof())`
- `void pbump (int __n)`

- virtual `pos_type seekoff` (`off_type __off`, `ios_base::seekdir __way`, `ios_base::openmode __mode=ios_base::in|ios_base::out`)
  - virtual `pos_type seekpos` (`pos_type __pos`, `ios_base::openmode __mode=ios_base::in|ios_base::out`)
  - virtual `__streambuf_type * setbuf` (`char_type *__s`, `streamsize __n`)
  - void `setg` (`char_type * __gbeg`, `char_type * __gnext`, `char_type * __gend`)
  - void `setp` (`char_type * __pbeg`, `char_type * __pend`)
  - virtual `streamsize showmanyc` ()
  - virtual `int sync` ()
  - virtual `int_type uflow` ()
  - virtual `int_type underflow` ()
  - virtual `streamsize xsgetn` (`char_type *__s`, `streamsize __n`)
  - virtual `streamsize xsputn` (`const char_type *__s`, `streamsize __n`)
- 
- `char_type * eback` () const
  - `char_type * gptr` () const
  - `char_type * egptr` () const
- 
- `char_type * pbase` () const
  - `char_type * pptr` () const
  - `char_type * epptr` () const

### Protected Attributes

- `char_type * _M_buf`
  - `bool _M_buf_allocated`
  - `size_t _M_buf_size`
  - `const __codecvt_type * _M_codecvt`
  - `char * _M_ext_buf`
  - `streamsize _M_ext_buf_size`
  - `char * _M_ext_end`
  - `const char * _M_ext_next`
  - `__file_type _M_file`
  - `__c_lock _M_lock`
  - `ios_base::openmode _M_mode`
  - `bool _M_reading`
  - `__state_type _M_state_beg`
  - `__state_type _M_state_cur`
  - `__state_type _M_state_last`
  - `bool _M_writing`
- 
- `char_type _M_pback`
  - `char_type * _M_pback_cur_save`
  - `char_type * _M_pback_end_save`
  - `bool _M_pback_init`

**Friends**

- `__gnu_cxx::__enable_if< __is_char<_CharT2>::__value, _CharT2 * >::__type __copy_move_a2` (`istreambuf_iterator<_CharT2>`, `istreambuf_iterator<_CharT2>`, `_CharT2 *`)
  - `streamsize __copy_streambufs_eof` (`__streambuf_type *`, `__streambuf_type *`, `bool &`)
  - class `basic_ios< char_type, traits_type >`
  - class `basic_istream< char_type, traits_type >`
  - class `basic_ostream< char_type, traits_type >`
  - `__gnu_cxx::__enable_if< __is_char<_CharT2>::__value, istreambuf_iterator<_CharT2> >::__type find` (`istreambuf_iterator<_CharT2>`, `istreambuf_iterator<_CharT2>`, `const _CharT2 &`)
  - `basic_istream<_CharT2, _Traits2> & getline` (`basic_istream<_CharT2, _Traits2> &`, `basic_string<_CharT2, _Traits2, _Alloc> &`, `_CharT2`)
  - class `ios_base`
  - class `istreambuf_iterator< char_type, traits_type >`
  - `basic_istream<_CharT2, _Traits2> & operator>>` (`basic_istream<_CharT2, _Traits2> &`, `_CharT2 *`)
  - `basic_istream<_CharT2, _Traits2> & operator>>` (`basic_istream<_CharT2, _Traits2> &`, `basic_string<_CharT2, _Traits2, _Alloc> &`)
  - class `ostreambuf_iterator< char_type, traits_type >`
- 
- locale `pubimbue` (`const locale &__loc`)
  - locale `getloc` () `const`
  - `__streambuf_type * pubsetbuf` (`char_type *__s`, `streamsize __n`)
  - `pos_type pubseekoff` (`off_type __off`, `ios_base::seekdir __way`, `ios_base::openmode __mode=ios_base::in|ios_base::out`)
  - `pos_type pubseekpos` (`pos_type __sp`, `ios_base::openmode __mode=ios_base::in|ios_base::out`)
  - `int pubsync` ()
  - `char_type * _M_in_beg`
  - `char_type * _M_in_cur`
  - `char_type * _M_in_end`
  - `char_type * _M_out_beg`
  - `char_type * _M_out_cur`
  - `char_type * _M_out_end`
  - locale `_M_buf_locale`

### 5.38.1 Detailed Description

`template<typename _CharT> class __gnu_cxx::enc_filebuf<_CharT>`

class `enc_filebuf`.

Definition at line 42 of file `enc_filebuf.h`.

### 5.38.2 Member Typedef Documentation

**5.38.2.1** `typedef basic_streambuf<char_type, traits_type> std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::__streambuf_type [inherited]`

This is a non-standard type.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

Definition at line 79 of file `fstream`.

**5.38.2.2** `typedef _CharT std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::char_type [inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

Definition at line 73 of file `fstream`.

**5.38.2.3** `typedef traits_type::int_type std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::int_type [inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

Definition at line 75 of file `fstream`.

#### 5.38.2.4 `typedef traits_type::off_type std::basic_filebuf<_CharT, encoding_char_traits<_CharT>::off_type [inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

Definition at line 77 of file `fstream`.

#### 5.38.2.5 `template<typename _CharT> typedef traits_type::pos_type __gnu_cxx::enc_filebuf<_CharT>::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`.

Definition at line 48 of file `enc_filebuf.h`.

#### 5.38.2.6 `template<typename _CharT> typedef encoding_char_traits<_CharT> __gnu_cxx::enc_filebuf<_CharT>::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`.

Definition at line 46 of file `enc_filebuf.h`.

### 5.38.3 Member Function Documentation

#### 5.38.3.1 `void std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_create_pback( ) [inline, protected, inherited]`

Initializes pback buffers, and moves normal buffers to safety. Assumptions: `_M_in_cur` has already been moved back

Definition at line 174 of file `fstream`.

**5.38.3.2** `void std::basic_filebuf<_CharT, encoding_char_traits<_CharT>  
>::_M_destroy_pback ( ) throw () [inline, protected,  
inherited]`

Deactivates pback buffer contents, and restores normal buffer. Assumptions: The pback buffer has only moved forward.

Definition at line 191 of file `fstream`.

**5.38.3.3** `void std::basic_filebuf<_CharT, encoding_char_traits<_CharT>  
>::_M_set_buffer ( streamsize __off ) [inline, protected,  
inherited]`

This function sets the pointers of the internal buffer, both get and put areas. Typically: `__off == egptr() - eback()` upon underflow/uflow (**read** mode); `__off == 0` upon overflow (**write** mode); `__off == -1` upon open, setbuf, seekoff/pos (**uncommitted** mode).

NB: `epptr() - pbase() == _M_buf_size - 1`, since `_M_buf_size` reflects the actual allocated memory and the last cell is reserved for the overflow char of a full put area.

Definition at line 392 of file `fstream`.

**5.38.3.4** `__filebuf_type* std::basic_filebuf<_CharT, encoding_char_traits<  
_CharT> >::close ( ) [inherited]`

Closes the currently associated file.

#### Returns

`this` on success, `NULL` on failure

If no file is currently open, this function immediately fails.

If a *put buffer area* exists, `overflow(eof)` is called to flush all the characters. The file is then closed.

If any operations fail, this function also fails.

**5.38.3.5** `char_type* std::basic_streambuf<_CharT, encoding_char_traits<  
_CharT> >::eback ( ) const [inline, protected,  
inherited]`

Access to the get area.



These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 462 of file `streambuf`.

**5.38.3.6** `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::egptr ( ) const` [`inline`, `protected`, `inherited`]

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 468 of file `streambuf`.

**5.38.3.7** `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::epptr ( ) const` [`inline`, `protected`, `inherited`]

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 515 of file `streambuf`.

**5.38.3.8** `void std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::gbump ( int __n ) [inline, protected, inherited]`

Moving the read position.

#### Parameters

*n* The delta by which to move.

This just advances the read position without returning any data.

Definition at line 478 of file streambuf.

**5.38.3.9** `locale std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::getloc ( ) const [inline, inherited]`

Locale access.

#### Returns

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 224 of file streambuf.

**5.38.3.10** `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::gptr ( ) const [inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 465 of file streambuf.

**5.38.3.11** `virtual void std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::imbue ( const locale & ) [protected, virtual, inherited]`

Changes translations.

#### Parameters

*loc* A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

#### Note

Base class version does nothing.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

**5.38.3.12** `streamsize std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::in_avail ( ) [inline, inherited]`

Looking ahead into the stream.

#### Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

Definition at line 264 of file `streambuf`.

**5.38.3.13** `bool std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::is_open ( ) const throw ( ) [inline, inherited]`

Returns true if the external file is open.

Definition at line 224 of file `fstream`.



**5.38.3.16** `virtual int_type std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::overflow ( int_type = _Traits::eof() )`  
[protected, virtual, inherited]

Consumes data from the buffer; writes to the controlled sequence.

#### Parameters

*c* An additional character to consume.

#### Returns

`eof()` to indicate failure, something else (usually *c*, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if *c* is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

#### Note

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

**5.38.3.17** `virtual int_type std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::pbackfail ( int_type = _Traits::eof() )`  
[protected, virtual, inherited]

Tries to back up the input sequence.

#### Parameters

*c* The character to be inserted back into the sequence.

#### Returns

`eof()` on failure, *some other value* on success

**Postcondition**

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

**5.38.3.18** `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pbase( ) const` [**inline, protected, inherited**]

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 509 of file `streambuf`.

**5.38.3.19** `void std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pbump( int __n )` [**inline, protected, inherited**]

Moving the write position.

**Parameters**

*n* The delta by which to move.

This just advances the write position without returning any data.

Definition at line 525 of file `streambuf`.

**5.38.3.20** `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pptr( ) const` [`inline`, `protected`, `inherited`]

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 512 of file `streambuf`.

**5.38.3.21** `locale std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pubimbue( const locale & __loc )` [`inline`, `inherited`]

Entry point for `imbue()`.

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 207 of file `streambuf`.

**5.38.3.22** `pos_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pubseekoff( off_type __off, ios_base::seekdir __way, ios_base::openmode __mode = ios_base::in | ios_base::out )` [`inline`, `inherited`]

Entry point for `imbue()`.

**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 241 of file `streambuf`.

**5.38.3.23** `pos_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pubseekpos ( pos_type __sp, ios_base::openmode __mode = ios_base::in | ios_base::out ) [inline, inherited]`

Entry point for `imbue()`.

**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 246 of file `streambuf`.

**5.38.3.24** `__streambuf_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pubsetbuf ( char_type * __s, streamsize __n ) [inline, inherited]`

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 237 of file `streambuf`.

**5.38.3.25** `int std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pubsync ( ) [inline, inherited]`



Entry point for `imbue()`.

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 251 of file `streambuf`.

#### 5.38.3.26 `int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT> >::sbumpc ( ) [inline, inherited]`

Getting the next character.

#### Returns

The next character, or `eof`.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 296 of file `streambuf`.

#### 5.38.3.27 `virtual pos_type std::basic_filebuf<_CharT, encoding_char_traits<_CharT> >::seekoff ( off_type, ios_base::seekdir, ios_base::openmode = ios_base::in | ios_base::out ) [protected, virtual, inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

#### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT> >`.

**5.38.3.28** `virtual pos_type std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::seekpos( pos_type, ios_base::openmode = ios_base::in | ios_base::out ) [protected, virtual, inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

#### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

**5.38.3.29** `virtual __streambuf_type* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::setbuf( char_type * __s, streamsize __n ) [protected, virtual, inherited]`

Manipulates the buffer.

#### Parameters

*s* Pointer to a buffer area.

*n* Size of *s*.

#### Returns

`this`

If no file has been opened, and both *s* and *n* are zero, then the stream becomes unbuffered. Otherwise, *s* is used as a buffer; see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

**5.38.3.30** `void std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::setg( char_type * __gbeg, char_type * __gnext, char_type * __gend ) [inline, protected, inherited]`

Setting the three read area pointers.

#### Parameters

*gbeg* A pointer.

*gnext* A pointer.

*gend* A pointer.

#### Postcondition

*gbeg* == `eback()`, *gnext* == `gptr()`, and *gend* == `egptr()`

Definition at line 489 of file `streambuf`.

**5.38.3.31** `void std::basic_streambuf<_CharT, encoding_char_traits<_CharT> >::setp( char_type * __pbeg, char_type * __pend ) [inline, protected, inherited]`

Setting the three write area pointers.

#### Parameters

*pbeg* A pointer.

*pend* A pointer.

#### Postcondition

*pbeg* == `pbase()`, *pbeg* == `pptr()`, and *pend* == `epptr()`

Definition at line 535 of file `streambuf`.

**5.38.3.32** `int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT> >::sgetc( ) [inline, inherited]`

Getting the next character.

#### Returns

The next character, or `eof`.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 318 of file `streambuf`.

**5.38.3.33** `streamsize std::basic_streambuf<_CharT, encoding_char_traits<_CharT> >::sgetn ( char_type * __s, streamsize __n )`  
`[inline, inherited]`

Entry point for `xsgetn`.

#### Parameters

*s* A buffer area.

*n* A count.

Returns `xsgetn(s,n)`. The effect is to fill `s[0]` through `s[n-1]` with characters from the input sequence, if possible.

Definition at line 337 of file `streambuf`.

**5.38.3.34** `virtual streamsize std::basic_filebuf<_CharT, encoding_char_traits<_CharT> >::showmanyc ( )` `[protected, virtual, inherited]`

Investigating the data available.

#### Returns

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.*  
 [27.5.2.4.3]/1

#### Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT> >`.

**5.38.3.35** `int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::snextc( ) [inline, inherited]`

Getting the next character.

**Returns**

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 278 of file `streambuf`.

**5.38.3.36** `int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::sputbackc( char_type __c ) [inline, inherited]`

Pushing characters back into the input stream.

**Parameters**

*c* The character to push back.

**Returns**

The previous character, if possible.

Similar to `sungetc()`, but *c* is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be *c*.

Definition at line 352 of file `streambuf`.

**5.38.3.37** `int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::sputc( char_type __c ) [inline, inherited]`

Entry point for all single-character output functions.

**Parameters**

*c* A character to output.

**Returns**

*c*, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores *c* in that position, increments the position, and returns `traits::to_int_type(c)`. If a write position is not available, returns `overflow(c)`.

Definition at line 404 of file `streambuf`.

**5.38.3.38** `streamsize std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::sputn ( const char_type * __s, streamsize __n )`  
`[inline, inherited]`

Entry point for all single-character output functions.

**Parameters**

*s* A buffer read area.

*n* A count.

One of two public output functions.

Returns `xputn(s,n)`. The effect is to write `s[0]` through `s[n-1]` to the output sequence, if possible.

Definition at line 430 of file `streambuf`.

**5.38.3.39** `int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::sungetc ( )` `[inline, inherited]`

Moving backwards in the input stream.

**Returns**

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbckfail()`. The effect is to *unget* the last character *gotten*.

Definition at line 377 of file `streambuf`.

**5.38.3.40** `virtual int std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::sync( void ) [protected, virtual, inherited]`

Synchronizes the buffer arrays with the controlled sequences.

#### Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

#### Note

Base class version does nothing, returns zero.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

**5.38.3.41** `virtual int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::uflow( ) [inline, protected, virtual, inherited]`

Fetches more data from the controlled sequence.

#### Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Definition at line 680 of file `streambuf`.

**5.38.3.42** `virtual int_type std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::underflow( ) [protected, virtual, inherited]`

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25.html>

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

**5.38.3.43** `virtual streamsize std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::xsgetn( char_type * __s, streamsize __n )`  
**[protected, virtual, inherited]**

Multiple character extraction.

**Parameters**

*s* A buffer area.

*n* Maximum number of characters to assign.

**Returns**

The number of characters assigned.

Fills `s[0]` through `s[n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either *n* characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.



**5.38.3.44** `virtual streamsize std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::xsputn ( const char_type * __s, streamsize __n )`  
`[protected, virtual, inherited]`

Multiple character insertion.

#### Parameters

- s* A buffer area.
- n* Maximum number of characters to write.

#### Returns

The number of characters written.

Writes *s*[0] through *s*[*n*-1] to the output sequence, as if by `sputc()`. Stops when either *n* characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

### 5.38.4 Member Data Documentation

**5.38.4.1** `char_type* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_buf` `[protected, inherited]`

Pointer to the beginning of internal buffer.

Definition at line 111 of file `fstream`.

**5.38.4.2** `locale std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_buf_locale` `[protected, inherited]`

Current locale setting.

Definition at line 190 of file `streambuf`.

#### 5.38.4.3 `size_t std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_buf_size` `[protected, inherited]`

Actual size of internal buffer. This number is equal to the size of the put area + 1 position, reserved for the overflow char of a full area.

Definition at line 118 of file `fstream`.

#### 5.38.4.4 `char* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_ext_buf` `[protected, inherited]`

Buffer for external characters. Used for input when `codecvt::always_noconv() == false`. When valid, this corresponds to `eback()`.

Definition at line 153 of file `fstream`.

#### 5.38.4.5 `streamsize std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_ext_buf_size` `[protected, inherited]`

Size of buffer held by `_M_ext_buf`.

Definition at line 158 of file `fstream`.

#### 5.38.4.6 `const char* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_ext_next` `[protected, inherited]`

Pointers into the buffer held by `_M_ext_buf` that delimit a subsequence of bytes that have been read but not yet converted. When valid, `_M_ext_next` corresponds to `egptr()`.

Definition at line 165 of file `fstream`.

#### 5.38.4.7 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_in_beg` `[protected, inherited]`

This is based on `_IO_FILE`, just reordered to be more consistent, and is intended to be the most minimal abstraction for an internal buffer.

- `get == input == read`
- `put == output == write`

Definition at line 182 of file `streambuf`.

#### 5.38.4.8 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_in_cur` `[protected, inherited]`

Entry point for `imbue()`.

##### Parameters

*loc* The new locale.

##### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 183 of file `streambuf`.

#### 5.38.4.9 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_in_end` `[protected, inherited]`

Entry point for `imbue()`.

##### Parameters

*loc* The new locale.

##### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 184 of file `streambuf`.

#### 5.38.4.10 `ios_base::openmode std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_mode` `[protected, inherited]`

Place to stash in || out || in | out settings for current filebuf.

Definition at line 96 of file `fstream`.

**5.38.4.11** `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_out_beg` `[protected, inherited]`

Entry point for `imbue()`.

**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 185 of file `streambuf`.

**5.38.4.12** `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_out_cur` `[protected, inherited]`

Entry point for `imbue()`.

**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 186 of file `streambuf`.

**5.38.4.13** `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_out_end` `[protected, inherited]`

Entry point for `imbue()`.

**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 187 of file `streambuf`.

**5.38.4.14** `char_type std::basic_filebuf<_CharT, encoding_char_traits<_CharT> >::_M_pback` `[protected, inherited]`

Necessary bits for putback buffer management.

**Note**

pbacks of over one character are not currently supported.

Definition at line 139 of file `fstream`.

**5.38.4.15** `char_type* std::basic_filebuf<_CharT, encoding_char_traits<_CharT> >::_M_pback_cur_save` `[protected, inherited]`

Necessary bits for putback buffer management.

**Note**

pbacks of over one character are not currently supported.

Definition at line 140 of file `fstream`.

**5.38.4.16** `char_type* std::basic_filebuf<_CharT, encoding_char_traits<_CharT> >::_M_pback_end_save` `[protected, inherited]`

Necessary bits for putback buffer management.

**Note**

pbacks of over one character are not currently supported.

Definition at line 141 of file `fstream`.

**5.38.4.17** `bool std::basic_filebuf<_CharT, encoding_char_traits<_CharT> >::_M_pback_init` `[protected, inherited]`

Necessary bits for putback buffer management.

## 5.39 `__gnu_cxx::encoding_char_traits<_CharT>` Struct Template Reference

### Note

pbacks of over one character are not currently supported.

Definition at line 142 of file `fstream`.

### 5.38.4.18 `bool std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_reading` [protected, inherited]

`_M_reading == false && _M_writing == false` for **uncommitted** mode; `_M_reading == true` for **read** mode; `_M_writing == true` for **write** mode;

NB: `_M_reading == true && _M_writing == true` is unused.

Definition at line 130 of file `fstream`.

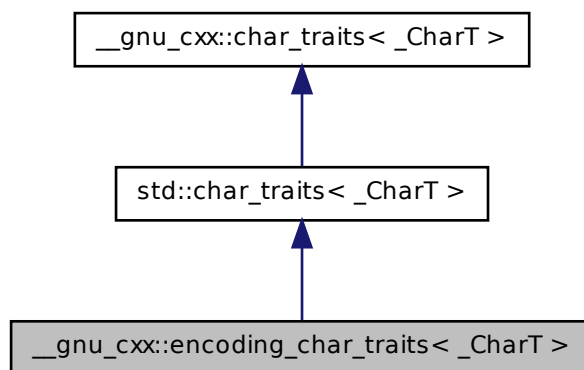
The documentation for this class was generated from the following file:

- [enc\\_filebuf.h](#)

## 5.39 `__gnu_cxx::encoding_char_traits<_CharT>` Struct Template Reference

[encoding\\_char\\_traits](#)

Inheritance diagram for `__gnu_cxx::encoding_char_traits<_CharT>`:



### Public Types

- typedef `_CharT` **char\_type**
- typedef `_Char_types<_CharT>::int_type` **int\_type**
- typedef `_Char_types<_CharT>::off_type` **off\_type**
- typedef `std::fpos<state_type>` **pos\_type**
- typedef `encoding_state` **state\_type**

### Static Public Member Functions

- static void **assign** (char\_type &\_\_c1, const char\_type &\_\_c2)
- static char\_type \* **assign** (char\_type \*\_\_s, std::size\_t \_\_n, char\_type \_\_a)
- static int **compare** (const char\_type \*\_\_s1, const char\_type \*\_\_s2, std::size\_t \_\_n)
- static char\_type \* **copy** (char\_type \*\_\_s1, const char\_type \*\_\_s2, std::size\_t \_\_n)
- static constexpr int\_type **eof** ()
- static constexpr bool **eq** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2)
- static const char\_type \* **find** (const char\_type \*\_\_s, std::size\_t \_\_n, const char\_type &\_\_a)
- static std::size\_t **length** (const char\_type \*\_\_s)
- static constexpr bool **lt** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static char\_type \* **move** (char\_type \*\_\_s1, const char\_type \*\_\_s2, std::size\_t \_\_n)
- static constexpr int\_type **not\_eof** (const int\_type &\_\_c)
- static constexpr char\_type **to\_char\_type** (const int\_type &\_\_c)
- static constexpr int\_type **to\_int\_type** (const char\_type &\_\_c)

#### 5.39.1 Detailed Description

`template<typename _CharT> struct __gnu_cxx::encoding_char_traits<_CharT>`

[encoding\\_char\\_traits](#)

Definition at line 212 of file `codecvt_specializations.h`.

The documentation for this struct was generated from the following file:

- [codecvt\\_specializations.h](#)

## 5.40 `__gnu_cxx::encoding_state` Class Reference

Extension to use `iconv` for dealing with character encodings.

### Public Types

- `typedef iconv_t descriptor_type`

### Public Member Functions

- `encoding_state` (`const char *__int`, `const char *__ext`, `int __ibom=0`, `int __ebom=0`, `int __bytes=1`)
- `encoding_state` (`const encoding_state &__obj`)
- `int character_ratio () const`
- `int external_bom () const`
- `const std::string external_encoding () const`
- `bool good () const throw ()`
- `const descriptor_type & in_descriptor () const`
- `int internal_bom () const`
- `const std::string internal_encoding () const`
- `encoding_state & operator= (const encoding_state &__obj)`
- `const descriptor_type & out_descriptor () const`

### Protected Member Functions

- `void construct (const encoding_state &__obj)`
- `void destroy () throw ()`
- `void init ()`

### Protected Attributes

- `int _M_bytes`
- `int _M_ext_bom`
- `std::string _M_ext_enc`
- `descriptor_type _M_in_desc`
- `int _M_int_bom`
- `std::string _M_int_enc`
- `descriptor_type _M_out_desc`



### 5.40.1 Detailed Description

Extension to use `iconv` for dealing with character encodings.

Definition at line 52 of file `codecvt_specializations.h`.

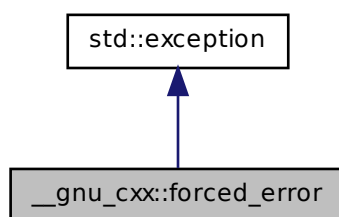
The documentation for this class was generated from the following file:

- [codecvt\\_specializations.h](#)

## 5.41 `__gnu_cxx::forced_error` Struct Reference

Thrown by exception safety machinery.

Inheritance diagram for `__gnu_cxx::forced_error`:



### Public Member Functions

- virtual `const char * what () const throw ()`

### 5.41.1 Detailed Description

Thrown by exception safety machinery.

Definition at line 75 of file `throw_allocator.h`.

### 5.41.2 Member Function Documentation

#### 5.41.2.1 virtual const char\* std::exception::what ( ) const throw () [virtual, inherited]

Returns a C-style character string describing the general cause of the current error.

Reimplemented in [std::bad\\_exception](#), [std::bad\\_alloc](#), [std::bad\\_cast](#), [std::bad\\_typeid](#), [std::future\\_error](#), [std::logic\\_error](#), [std::runtime\\_error](#), [std::ios\\_base::failure](#), and [std::bad\\_weak\\_ptr](#).

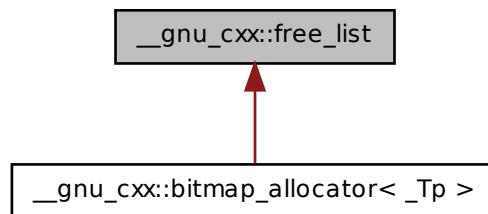
The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.42 \_\_gnu\_cxx::free\_list Class Reference

The free list class for managing chunks of memory to be given to and returned by the [bitmap\\_allocator](#).

Inheritance diagram for \_\_gnu\_cxx::free\_list:



### Public Types

- typedef \_\_mutex **\_\_mutex\_type**
- typedef vector\_type::iterator **iterator**
- typedef size\_t \* **value\_type**
- typedef [\\_\\_detail::\\_\\_mini\\_vector](#)< value\_type > **vector\_type**

### Public Member Functions

- void `_M_clear` ()
- `size_t * _M_get` (size\_t `__sz`) throw (std::bad\_alloc)
- void `_M_insert` (size\_t \* `__addr`) throw ()

#### 5.42.1 Detailed Description

The free list class for managing chunks of memory to be given to and returned by the [bitmap\\_allocator](#).

Definition at line 525 of file `bitmap_allocator.h`.

#### 5.42.2 Member Function Documentation

##### 5.42.2.1 void `__gnu_cxx::free_list::_M_clear` ( )

This function just clears the internal Free List, and gives back all the memory to the OS.

##### 5.42.2.2 `size_t * __gnu_cxx::free_list::_M_get` ( `size_t __sz` ) throw (std::bad\_alloc)

This function gets a block of memory of the specified size from the free list.

#### Parameters

`__sz` The size in bytes of the memory required.

#### Returns

A pointer to the new memory block of size at least equal to that requested.

##### 5.42.2.3 void `__gnu_cxx::free_list::_M_insert` ( `size_t * __addr` ) throw () [inline]

This function returns the block of memory to the internal free list.

### 5.43 `__gnu_cxx::hash_map<_Key, _Tp, _HashFn, _EqualKey, _Alloc>` Class Template Reference 961

---

#### Parameters

`__addr` The pointer to the memory block that was given by a call to the `_M_get` function.

Definition at line 635 of file `bitmap_allocator.h`.

Referenced by `__gnu_cxx::bitmap_allocator<_Tp>::_M_deallocate_single_object()`.

The documentation for this class was generated from the following file:

- [bitmap\\_allocator.h](#)

### 5.43 `__gnu_cxx::hash_map<_Key, _Tp, _HashFn, _EqualKey, _Alloc>` Class Template Reference

#### Public Types

- `typedef _Ht::allocator_type allocator_type`
- `typedef _Ht::const_iterator const_iterator`
- `typedef _Ht::const_pointer const_pointer`
- `typedef _Ht::const_reference const_reference`
- `typedef _Tp data_type`
- `typedef _Ht::difference_type difference_type`
- `typedef _Ht::hasher hasher`
- `typedef _Ht::iterator iterator`
- `typedef _Ht::key_equal key_equal`
- `typedef _Ht::key_type key_type`
- `typedef _Tp mapped_type`
- `typedef _Ht::pointer pointer`
- `typedef _Ht::reference reference`
- `typedef _Ht::size_type size_type`
- `typedef _Ht::value_type value_type`

#### Public Member Functions

- `hash_map (size_type __n)`
- `template<class _InputIterator>  
hash_map (_InputIterator __f, _InputIterator __l)`
- `template<class _InputIterator>  
hash_map (_InputIterator __f, _InputIterator __l, size_type __n)`
- `template<class _InputIterator>  
hash_map (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf)`

- `template<class _InputIterator >`  
   **hash\_map** (`_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type()`)
- **hash\_map** (`size_type __n, const hasher &__hf`)
- **hash\_map** (`size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type()`)
- iterator **begin** ()
- `const_iterator begin` () `const`
- `size_type bucket_count` () `const`
- `void clear` ()
- `size_type count` (`const key_type &__key`) `const`
- `size_type elems_in_bucket` (`size_type __n`) `const`
- `bool empty` () `const`
- iterator **end** ()
- `const_iterator end` () `const`
- `pair< iterator, iterator > equal_range` (`const key_type &__key`)
- `pair< const_iterator, const_iterator > equal_range` (`const key_type &__key`) `const`
- `size_type erase` (`const key_type &__key`)
- `void erase` (`iterator __f, iterator __l`)
- `void erase` (`iterator __it`)
- iterator **find** (`const key_type &__key`)
- `const_iterator find` (`const key_type &__key`) `const`
- `allocator_type get_allocator` () `const`
- hasher **hash\_func** () `const`
- `template<class _InputIterator >`  
   `void insert` (`_InputIterator __f, _InputIterator __l`)
- `pair< iterator, bool > insert` (`const value_type &__obj`)
- `pair< iterator, bool > insert_noresize` (`const value_type &__obj`)
- `key_equal key_eq` () `const`
- `size_type max_bucket_count` () `const`
- `size_type max_size` () `const`
- `_Tp & operator[]` (`const key_type &__key`)
- `void resize` (`size_type __hint`)
- `size_type size` () `const`
- `void swap` (`hash_map &__hs`)

## Friends

- `template<class _K1, class _T1, class _HF, class _EqK, class _AI >`  
   `bool operator==` (`const hash_map< _K1, _T1, _HF, _EqK, _AI > &, const hash_map< _K1, _T1, _HF, _EqK, _AI > &`)

#### 5.43.1 Detailed Description

```
template<class _Key, class _Tp, class _HashFn = hash<_Key>, class _EqualKey
= equal_to<_Key>, class _Alloc = allocator<_Tp>> class __gnu_cxx::hash_
map< _Key, _Tp, _HashFn, _EqualKey, _Alloc >
```

This is an SGI extension.

#### Todo

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

Definition at line 77 of file `hash_map`.

The documentation for this class was generated from the following file:

- [hash\\_map](#)

#### 5.44 `__gnu_cxx::hash_multimap<_Key, _Tp, _HashFn, _EqualKey, _Alloc>` Class Template Reference

##### Public Types

- `typedef _Ht::allocator_type allocator_type`
- `typedef _Ht::const_iterator const_iterator`
- `typedef _Ht::const_pointer const_pointer`
- `typedef _Ht::const_reference const_reference`
- `typedef _Tp data_type`
- `typedef _Ht::difference_type difference_type`
- `typedef _Ht::hasher hasher`
- `typedef _Ht::iterator iterator`
- `typedef _Ht::key_equal key_equal`
- `typedef _Ht::key_type key_type`
- `typedef _Tp mapped_type`
- `typedef _Ht::pointer pointer`
- `typedef _Ht::reference reference`
- `typedef _Ht::size_type size_type`
- `typedef _Ht::value_type value_type`

##### Public Member Functions

- `hash_multimap (size_type __n)`

- `template<class _InputIterator >`  
`hash_multimap` (`_InputIterator __f, _InputIterator __l`)
- `template<class _InputIterator >`  
`hash_multimap` (`_InputIterator __f, _InputIterator __l, size_type __n`)
- `template<class _InputIterator >`  
`hash_multimap` (`_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf`)
- `template<class _InputIterator >`  
`hash_multimap` (`_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type()`)
- `hash_multimap` (`size_type __n, const hasher &__hf`)
- `hash_multimap` (`size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type()`)
- iterator `begin` ()
- `const_iterator begin` () `const`
- `size_type bucket_count` () `const`
- `void clear` ()
- `size_type count` (`const key_type &__key`) `const`
- `size_type elems_in_bucket` (`size_type __n`) `const`
- `bool empty` () `const`
- iterator `end` ()
- `const_iterator end` () `const`
- `pair< const_iterator, const_iterator > equal_range` (`const key_type &__key`) `const`
- `pair< iterator, iterator > equal_range` (`const key_type &__key`)
- `size_type erase` (`const key_type &__key`)
- `void erase` (`iterator __f, iterator __l`)
- `void erase` (`iterator __it`)
- iterator `find` (`const key_type &__key`)
- `const_iterator find` (`const key_type &__key`) `const`
- `allocator_type get_allocator` () `const`
- hasher `hash_funct` () `const`
- `template<class _InputIterator >`  
`void insert` (`_InputIterator __f, _InputIterator __l`)
- iterator `insert` (`const value_type &__obj`)
- iterator `insert_noresize` (`const value_type &__obj`)
- `key_equal key_eq` () `const`
- `size_type max_bucket_count` () `const`
- `size_type max_size` () `const`
- `void resize` (`size_type __hint`)
- `size_type size` () `const`
- `void swap` (`hash_multimap &__hs`)

## Friends

- `template<class _K1, class _T1, class _HF, class _EqK, class _Al>`  
`bool operator== (const hash\_multimap<_K1, _T1, _HF, _EqK, _Al> &, const`  
`hash\_multimap<_K1, _T1, _HF, _EqK, _Al> &)`

### 5.44.1 Detailed Description

`template<class _Key, class _Tp, class _HashFn = hash<_Key>, class _EqualKey`  
`= equal_to<_Key>, class _Alloc = allocator<_Tp>> class __gnu_cxx::hash-`  
`multimap<_Key, _Tp, _HashFn, _EqualKey, _Alloc>`

This is an SGI extension.

## Todo

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

Definition at line 290 of file `hash_map`.

The documentation for this class was generated from the following file:

- [hash\\_map](#)

## 5.45 `__gnu_cxx::hash_multiset<_Value, _HashFcn, _EqualKey, _Alloc>` Class Template Reference

### Public Types

- `typedef _Ht::allocator_type allocator_type`
- `typedef _Ht::const_iterator const_iterator`
- `typedef _Alloc::const_pointer const_pointer`
- `typedef _Alloc::const_reference const_reference`
- `typedef _Ht::difference_type difference_type`
- `typedef _Ht::hasher hasher`
- `typedef _Ht::const_iterator iterator`
- `typedef _Ht::key_equal key_equal`
- `typedef _Ht::key_type key_type`
- `typedef _Alloc::pointer pointer`
- `typedef _Alloc::reference reference`
- `typedef _Ht::size_type size_type`
- `typedef _Ht::value_type value_type`



### Public Member Functions

- **hash\_multiset** (size\_type \_\_n)
- template<class \_InputIterator >  
**hash\_multiset** (\_InputIterator \_\_f, \_InputIterator \_\_l)
- template<class \_InputIterator >  
**hash\_multiset** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n)
- template<class \_InputIterator >  
**hash\_multiset** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n, const hasher &\_\_hf)
- template<class \_InputIterator >  
**hash\_multiset** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n, const hasher &\_\_hf, const key\_equal &\_\_eq, const allocator\_type &\_\_a=allocator\_type())
- **hash\_multiset** (size\_type \_\_n, const hasher &\_\_hf)
- **hash\_multiset** (size\_type \_\_n, const hasher &\_\_hf, const key\_equal &\_\_eq, const allocator\_type &\_\_a=allocator\_type())
- iterator **begin** () const
- size\_type **bucket\_count** () const
- void **clear** ()
- size\_type **count** (const key\_type &\_\_key) const
- size\_type **elems\_in\_bucket** (size\_type \_\_n) const
- bool **empty** () const
- iterator **end** () const
- pair< iterator, iterator > **equal\_range** (const key\_type &\_\_key) const
- void **erase** (iterator \_\_f, iterator \_\_l)
- void **erase** (iterator \_\_it)
- size\_type **erase** (const key\_type &\_\_key)
- iterator **find** (const key\_type &\_\_key) const
- allocator\_type **get\_allocator** () const
- hasher **hash\_funct** () const
- template<class \_InputIterator >  
void **insert** (\_InputIterator \_\_f, \_InputIterator \_\_l)
- iterator **insert** (const value\_type &\_\_obj)
- iterator **insert\_noresize** (const value\_type &\_\_obj)
- key\_equal **key\_eq** () const
- size\_type **max\_bucket\_count** () const
- size\_type **max\_size** () const
- void **resize** (size\_type \_\_hint)
- size\_type **size** () const
- void **swap** ([hash\\_multiset](#) &hs)

## Friends

- `template<class _Val, class _HF, class _EqK, class _Al>`  
`bool operator== (const hash\_multiset< _Val, _HF, _EqK, _Al > &, const`  
`hash\_multiset< _Val, _HF, _EqK, _Al > &)`

### 5.45.1 Detailed Description

`template<class _Value, class _HashFcn = hash<_Value>, class _EqualKey =`  
`equal_to<_Value>, class _Alloc = allocator<_Value>> class __gnu_cxx::hash-`  
`multiset< _Value, _HashFcn, _EqualKey, _Alloc >`

This is an SGI extension.

## Todo

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

Definition at line 286 of file `hash_set`.

The documentation for this class was generated from the following file:

- [hash\\_set](#)

## 5.46 `__gnu_cxx::hash_set<_Value, _HashFcn, _EqualKey, _Alloc>` > Class Template Reference

### Public Types

- `typedef _Ht::allocator_type allocator_type`
- `typedef _Ht::const_iterator const_iterator`
- `typedef _Alloc::const_pointer const_pointer`
- `typedef _Alloc::const_reference const_reference`
- `typedef _Ht::difference_type difference_type`
- `typedef _Ht::hasher hasher`
- `typedef _Ht::const_iterator iterator`
- `typedef _Ht::key_equal key_equal`
- `typedef _Ht::key_type key_type`
- `typedef _Alloc::pointer pointer`
- `typedef _Alloc::reference reference`
- `typedef _Ht::size_type size_type`
- `typedef _Ht::value_type value_type`

### Public Member Functions

- **hash\_set** (size\_type \_\_n)
- template<class \_InputIterator >  
**hash\_set** (\_InputIterator \_\_f, \_InputIterator \_\_l)
- template<class \_InputIterator >  
**hash\_set** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n)
- template<class \_InputIterator >  
**hash\_set** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n, const hasher &\_  
\_hf)
- template<class \_InputIterator >  
**hash\_set** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n, const hasher &\_  
\_hf, const key\_equal &\_\_eq, const allocator\_type &\_\_a=allocator\_type())
- **hash\_set** (size\_type \_\_n, const hasher &\_\_hf)
- **hash\_set** (size\_type \_\_n, const hasher &\_\_hf, const key\_equal &\_\_eq, const  
allocator\_type &\_\_a=allocator\_type())
- iterator **begin** () const
- size\_type **bucket\_count** () const
- void **clear** ()
- size\_type **count** (const key\_type &\_\_key) const
- size\_type **elems\_in\_bucket** (size\_type \_\_n) const
- bool **empty** () const
- iterator **end** () const
- pair< iterator, iterator > **equal\_range** (const key\_type &\_\_key) const
- void **erase** (iterator \_\_f, iterator \_\_l)
- void **erase** (iterator \_\_it)
- size\_type **erase** (const key\_type &\_\_key)
- iterator **find** (const key\_type &\_\_key) const
- allocator\_type **get\_allocator** () const
- hasher **hash\_funct** () const
- template<class \_InputIterator >  
void **insert** (\_InputIterator \_\_f, \_InputIterator \_\_l)
- pair< iterator, bool > **insert** (const value\_type &\_\_obj)
- pair< iterator, bool > **insert\_noresize** (const value\_type &\_\_obj)
- key\_equal **key\_eq** () const
- size\_type **max\_bucket\_count** () const
- size\_type **max\_size** () const
- void **resize** (size\_type \_\_hint)
- size\_type **size** () const
- void **swap** (hash\_set &\_\_hs)

## Friends

- `template<class _Val, class _HF, class _EqK, class _Al >`  
`bool operator== (const hash\_set< _Val, _HF, _EqK, _Al > &, const hash\_set< _Val, _HF, _EqK, _Al > &)`

### 5.46.1 Detailed Description

`template<class _Value, class _HashFcn = hash<_Value>, class _EqualKey = equal\_to<_Value>, class _Alloc = allocator<_Value>> class __gnu_cxx::hash_set< _Value, _HashFcn, _EqualKey, _Alloc >`

This is an SGI extension.

## Todo

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

Definition at line 85 of file `hash_set`.

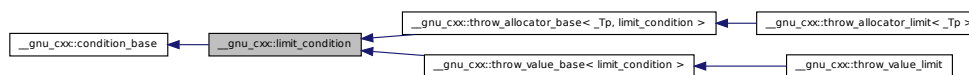
The documentation for this class was generated from the following file:

- [hash\\_set](#)

## 5.47 `__gnu_cxx::limit_condition` Struct Reference

Base class for incremental control and throw.

Inheritance diagram for `__gnu_cxx::limit_condition`:



## Classes

- struct [always\\_adjustor](#)  
*Always enter the condition.*
- struct [limit\\_adjustor](#)

*Enter the  $n$ th condition.*

- struct `never_adjustor`

*Never enter the condition.*

### Static Public Member Functions

- static `size_t & count ()`
- static `size_t & limit ()`
- static void `set_limit (const size_t __l)`
- static void `throw_conditionally ()`

#### 5.47.1 Detailed Description

Base class for incremental control and throw.

Definition at line 264 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.48 `__gnu_cxx::limit_condition::always_adjustor` Struct Reference

Always enter the condition.

Inherits `__gnu_cxx::limit_condition::adjustor_base`.

#### 5.48.1 Detailed Description

Always enter the condition.

Definition at line 288 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.49 `__gnu_cxx::limit_condition::limit_adjustor` Struct Reference

Enter the  $n$ th condition.

Inherits `__gnu_cxx::limit_condition::adjustor_base`.

### Public Member Functions

- `limit_adjustor` (const size\_t \_\_l)

#### 5.49.1 Detailed Description

Enter the nth condition.

Definition at line 294 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

### 5.50 `__gnu_cxx::limit_condition::never_adjustor` Struct Reference

Never enter the condition.

Inherits `__gnu_cxx::limit_condition::adjustor_base`.

#### 5.50.1 Detailed Description

Never enter the condition.

Definition at line 282 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

### 5.51 `__gnu_cxx::malloc_allocator< _Tp >` Class Template Reference

An allocator that uses `malloc`.

This is precisely the allocator defined in the C++ Standard.

- all allocation calls `malloc`
- all deallocation calls `free`.

### Public Types

- `typedef const _Tp * const_pointer`

- `typedef const _Tp & const_reference`
- `typedef ptrdiff_t difference_type`
- `typedef _Tp * pointer`
- `typedef _Tp & reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

### Public Member Functions

- `malloc_allocator` (const [malloc\\_allocator](#) &) throw ()
- `template<typename _Tp1 > malloc_allocator` (const [malloc\\_allocator](#)<\_Tp1> &) throw ()
- `pointer address` (reference \_\_x) const
- `const_pointer address` (const\_reference \_\_x) const
- `pointer allocate` (size\_type \_\_n, const void \* = 0)
- `void construct` (pointer \_\_p, const \_Tp & \_\_val)
- `template<typename... _Args> void construct` (pointer \_\_p, \_Args &&... \_\_args)
- `void deallocate` (pointer \_\_p, size\_type)
- `void destroy` (pointer \_\_p)
- `size_type max_size` () const throw ()

#### 5.51.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::malloc_allocator<_Tp>`

An allocator that uses malloc.

This is precisely the allocator defined in the C++ Standard.

- all allocation calls malloc
- all deallocation calls free.

Definition at line 54 of file `malloc_allocator.h`.

The documentation for this class was generated from the following file:

- [malloc\\_allocator.h](#)

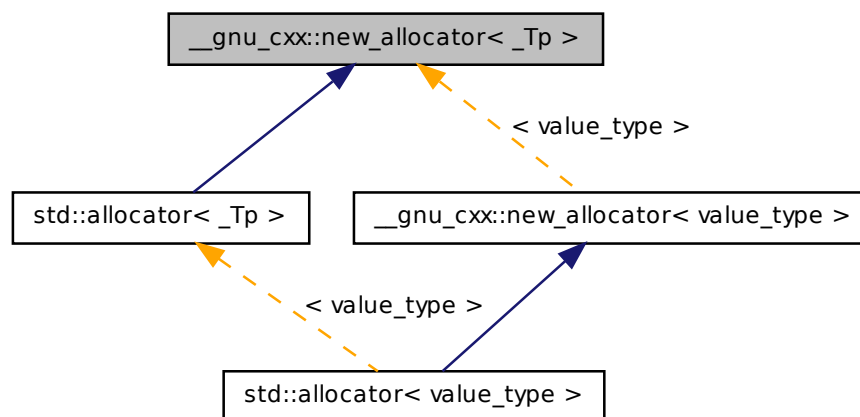
## 5.52 `__gnu_cxx::new_allocator<_Tp>` Class Template Reference

An allocator that uses global new, as per [20.4].

This is precisely the allocator defined in the C++ Standard.

- all allocation calls operator new
- all deallocation calls operator delete.

Inheritance diagram for `__gnu_cxx::new_allocator<_Tp>`:



### Public Types

- `typedef const _Tp * const_pointer`
- `typedef const _Tp & const_reference`
- `typedef ptrdiff_t difference_type`
- `typedef _Tp * pointer`
- `typedef _Tp & reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`



**Public Member Functions**

- **new\_allocator** (const [new\\_allocator](#) &) throw ()
- `template<typename _Tp1 >`  
**new\_allocator** (const [new\\_allocator](#)< \_Tp1 > &) throw ()
- pointer **address** (reference \_\_x) const
- `const_pointer` **address** (const\_reference \_\_x) const
- pointer **allocate** (size\_type \_\_n, const void \*=0)
- void **construct** (pointer \_\_p, const \_Tp &\_\_val)
- `template<typename... _Args>`  
void **construct** (pointer \_\_p, \_Args &&...\_\_args)
- void **deallocate** (pointer \_\_p, size\_type)
- void **destroy** (pointer \_\_p)
- size\_type **max\_size** () const throw ()

**5.52.1 Detailed Description**

`template<typename _Tp> class __gnu_cxx::new_allocator< _Tp >`

An allocator that uses global new, as per [20.4].

This is precisely the allocator defined in the C++ Standard.

- all allocation calls operator new
- all deallocation calls operator delete.

Definition at line 54 of file `new_allocator.h`.

The documentation for this class was generated from the following file:

- [new\\_allocator.h](#)

**5.53 `__gnu_cxx::project1st< _Arg1, _Arg2 >` Struct Template Reference**

An [SGI extension](#) .

Inherits `__gnu_cxx::_Project1st< _Arg1, _Arg2 >`.

**Public Types**

- typedef \_Arg1 [first\\_argument\\_type](#)
- typedef \_Result [result\\_type](#)
- typedef \_Arg2 [second\\_argument\\_type](#)

**Public Member Functions**

- `_Arg1 operator() (const _Arg1 &__x, const _Arg2 &) const`

**5.53.1 Detailed Description**

`template<class _Arg1, class _Arg2> struct __gnu_cxx::project1st< _Arg1, _Arg2 >`

An [SGI extension](#) .

Definition at line 234 of file `ext/functional`.

**5.53.2 Member Typedef Documentation**

**5.53.2.1** `template<typename _Arg1, typename _Arg2, typename _Result>  
typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result  
>::first_argument_type [inherited]`

`first_argument_type` is the type of the first argument

Definition at line 118 of file `stl_function.h`.

**5.53.2.2** `template<typename _Arg1, typename _Arg2, typename _Result>  
typedef _Result std::binary_function< _Arg1, _Arg2, _Result  
>::result_type [inherited]`

`result_type` is the return type

Definition at line 124 of file `stl_function.h`.

**5.53.2.3** `template<typename _Arg1, typename _Arg2, typename _Result>  
typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result  
>::second_argument_type [inherited]`

`second_argument_type` is the type of the second argument

Definition at line 121 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

## 5.54 `__gnu_cxx::project2nd< _Arg1, _Arg2 >` Struct Template Reference 976

- [ext/functional](#)

### 5.54 `__gnu_cxx::project2nd< _Arg1, _Arg2 >` Struct Template Reference

An [SGI extension](#).

Inherits `__gnu_cxx::Project2nd< _Arg1, _Arg2 >`.

#### Public Types

- typedef `_Arg1` [first\\_argument\\_type](#)
- typedef `_Result` [result\\_type](#)
- typedef `_Arg2` [second\\_argument\\_type](#)

#### Public Member Functions

- `_Arg2 operator()` (`const _Arg1 &`, `const _Arg2 &__y`) `const`

##### 5.54.1 Detailed Description

```
template<class _Arg1, class _Arg2> struct __gnu_cxx::project2nd< _Arg1, _Arg2 >
```

An [SGI extension](#).

Definition at line 238 of file `ext/functional`.

##### 5.54.2 Member Typedef Documentation

```
5.54.2.1 template<typename _Arg1, typename _Arg2, typename _Result>
 typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result
 >::first_argument_type [inherited]
```

`first_argument_type` is the type of the first argument

Definition at line 118 of file `stl_function.h`.

**5.54.2.2** `template<typename _Arg1, typename _Arg2, typename _Result>`  
`typedef _Result std::binary_function< _Arg1, _Arg2, _Result`  
`>::result_type [inherited]`

`result_type` is the return type

Definition at line 124 of file `stl_function.h`.

**5.54.2.3** `template<typename _Arg1, typename _Arg2, typename _Result>`  
`typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result`  
`>::second_argument_type [inherited]`

`second_argument_type` is the type of the second argument

Definition at line 121 of file `stl_function.h`.

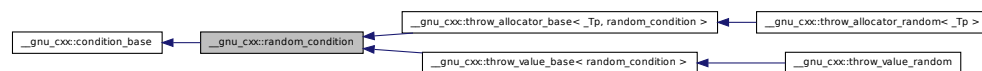
The documentation for this struct was generated from the following file:

- [ext/functional](#)

## 5.55 `__gnu_cxx::random_condition` Struct Reference

Base class for random probability control and throw.

Inheritance diagram for `__gnu_cxx::random_condition`:



### Classes

- struct [always\\_adjustor](#)  
*Always enter the condition.*
- struct [group\\_adjustor](#)  
*Group condition.*
- struct [never\\_adjustor](#)

*Never enter the condition.*

#### Public Member Functions

- void **seed** (unsigned long \_\_s)

#### Static Public Member Functions

- static void **set\_probability** (double \_\_p)
- static void **throw\_conditionally** ()

##### 5.55.1 Detailed Description

Base class for random probability control and throw.

Definition at line 336 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.56 `__gnu_cxx::random_condition::always_adjustor` Struct Reference

Always enter the condition.

Inherits `__gnu_cxx::random_condition::adjustor_base`.

##### 5.56.1 Detailed Description

Always enter the condition.

Definition at line 369 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.57 `__gnu_cxx::random_condition::group_adjustor` Struct Reference

Group condition.

Inherits `__gnu_cxx::random_condition::adjustor_base`.

## Public Member Functions

- `group_adjustor` (size\_t size)

### 5.57.1 Detailed Description

Group condition.

Definition at line 354 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.58 `__gnu_cxx::random_condition::never_adjustor` Struct Reference

Never enter the condition.

Inherits `__gnu_cxx::random_condition::adjustor_base`.

### 5.58.1 Detailed Description

Never enter the condition.

Definition at line 363 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.59 `__gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >` Struct Template Reference

Inherits `std::_Rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >`.

## Public Types

- `typedef _Rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc > _Base`
- `typedef const _Rb_tree_node< _Val > * _Const_Link_type`
- `typedef _Rb_tree_node< _Val > * _Link_type`
- `typedef _Base::allocator_type allocator_type`
- `typedef _Rb_tree_const_iterator< value_type > const_iterator`
- `typedef const value_type * const_pointer`

- typedef const value\_type & **const\_reference**
- typedef `std::reverse_iterator`< const\_iterator > **const\_reverse\_iterator**
- typedef ptrdiff\_t **difference\_type**
- typedef \_Rb\_tree\_iterator< value\_type > **iterator**
- typedef \_Key **key\_type**
- typedef value\_type \* **pointer**
- typedef value\_type & **reference**
- typedef `std::reverse_iterator`< iterator > **reverse\_iterator**
- typedef size\_t **size\_type**
- typedef \_Val **value\_type**

### Public Member Functions

- **rb\_tree** (const \_Compare &\_\_comp=\_Compare(), const allocator\_type &\_\_a=allocator\_type())
- bool **\_\_rb\_verify** () const
- const \_Node\_allocator & **\_M\_get\_Node\_allocator** () const
- \_Node\_allocator & **\_M\_get\_Node\_allocator** ()
- template<typename \_Arg >  
iterator **\_M\_insert\_equal** (\_Arg &&\_\_x)
- template<typename \_InputIterator >  
void **\_M\_insert\_equal** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<class \_II >  
void **\_M\_insert\_equal** (\_II \_\_first, \_II \_\_last)
- template<typename \_Arg >  
iterator **\_M\_insert\_equal\_** (const\_iterator \_\_position, \_Arg &&\_\_x)
- template<typename \_Arg >  
`pair`< iterator, bool > **\_M\_insert\_unique** (\_Arg &&\_\_x)
- template<class \_II >  
void **\_M\_insert\_unique** (\_II \_\_first, \_II \_\_last)
- template<typename \_InputIterator >  
void **\_M\_insert\_unique** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_Arg >  
iterator **\_M\_insert\_unique\_** (const\_iterator \_\_position, \_Arg &&\_\_x)
- const\_iterator **begin** () const
- iterator **begin** ()
- void **clear** ()
- size\_type **count** (const key\_type &\_\_k) const
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- `pair`< iterator, iterator > **equal\_range** (const key\_type &\_\_k)

- `pair< const_iterator, const_iterator > equal_range (const key_type &__k) const`
- iterator `erase` (const\_iterator \_\_first, const\_iterator \_\_last)
- iterator `erase` (const\_iterator \_\_position)
- size\_type `erase` (const key\_type &\_\_x)
- void `erase` (const key\_type \*\_\_first, const key\_type \*\_\_last)
- iterator `find` (const key\_type &\_\_k)
- const\_iterator `find` (const key\_type &\_\_k) const
- allocator\_type `get_allocator` () const
- \_Compare `key_comp` () const
- iterator `lower_bound` (const key\_type &\_\_k)
- const\_iterator `lower_bound` (const key\_type &\_\_k) const
- size\_type `max_size` () const
- reverse\_iterator `rbegin` ()
- const\_reverse\_iterator `rbegin` () const
- reverse\_iterator `rend` ()
- const\_reverse\_iterator `rend` () const
- size\_type `size` () const
- void `swap` (\_Rb\_tree &\_\_t)
- iterator `upper_bound` (const key\_type &\_\_k)
- const\_iterator `upper_bound` (const key\_type &\_\_k) const

### Protected Types

- typedef `_Rb_tree_node_base * _Base_ptr`
- typedef `const _Rb_tree_node_base * _Const_Base_ptr`

### Protected Member Functions

- `_Link_type _M_begin` ()
- `_Const_Link_type _M_begin` () const
- `_Link_type _M_clone_node` (\_Const\_Link\_type \_\_x)
- `template<typename... _Args> _Link_type _M_create_node` (\_Args &&...\_\_args)
- `void _M_destroy_node` (\_Link\_type \_\_p)
- `_Const_Link_type _M_end` () const
- `_Link_type _M_end` ()
- `_Link_type _M_get_node` ()
- `_Base_ptr & _M_leftmost` ()
- `_Const_Base_ptr _M_leftmost` () const
- `void _M_put_node` (\_Link\_type \_\_p)
- `_Base_ptr & _M_rightmost` ()



- `_Const_Base_ptr _M_rightmost () const`
- `_Base_ptr & _M_root ()`
- `_Const_Base_ptr _M_root () const`

#### Static Protected Member Functions

- `static const _Key & _S_key (_Const_Link_type __x)`
- `static const _Key & _S_key (_Const_Base_ptr __x)`
- `static _Link_type _S_left (_Base_ptr __x)`
- `static _Const_Link_type _S_left (_Const_Base_ptr __x)`
- `static _Base_ptr _S_maximum (_Base_ptr __x)`
- `static _Const_Base_ptr _S_maximum (_Const_Base_ptr __x)`
- `static _Base_ptr _S_minimum (_Base_ptr __x)`
- `static _Const_Base_ptr _S_minimum (_Const_Base_ptr __x)`
- `static _Const_Link_type _S_right (_Const_Base_ptr __x)`
- `static _Link_type _S_right (_Base_ptr __x)`
- `static const_reference _S_value (_Const_Link_type __x)`
- `static const_reference _S_value (_Const_Base_ptr __x)`

#### Protected Attributes

- `_Rb_tree_impl< _Compare > _M_impl`

#### 5.59.1 Detailed Description

`template<class _Key, class _Value, class _KeyOfValue, class _Compare, class _Alloc = allocator<_Value>> struct __gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >`

This is an SGI extension.

#### Todo

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

Definition at line 80 of file `rb_tree`.

The documentation for this struct was generated from the following file:

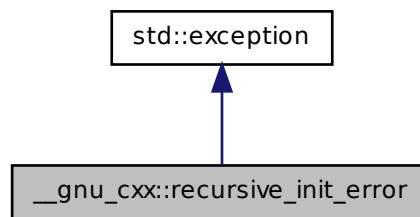
- [rb\\_tree](#)

## 5.60 `__gnu_cxx::recursive_init_error` Class Reference

Exception thrown by `__cxa_guard_acquire`.

6.7[stmt.dcl]/4: If control re-enters the declaration (recursively) while the object is being initialized, the behavior is undefined.

Inheritance diagram for `__gnu_cxx::recursive_init_error`:



### Public Member Functions

- virtual const char \* `what` () const throw ()

#### 5.60.1 Detailed Description

Exception thrown by `__cxa_guard_acquire`.

6.7[stmt.dcl]/4: If control re-enters the declaration (recursively) while the object is being initialized, the behavior is undefined. Since we already have a library function to handle locking, we might as well check for this situation and throw an exception. We use the second byte of the guard variable to remember that we're in the middle of an initialization.

Definition at line 615 of file `cxxabi.h`.

#### 5.60.2 Member Function Documentation

##### 5.60.2.1 virtual const char\* `std::exception::what` ( ) const throw () [virtual, inherited]

Returns a C-style character string describing the general cause of the current error.

Reimplemented in `std::bad_exception`, `std::bad_alloc`, `std::bad_cast`, `std::bad_typeid`, `std::future_error`, `std::logic_error`, `std::runtime_error`, `std::ios_base::failure`, and `std::bad_weak_ptr`.

The documentation for this class was generated from the following file:

- [cxxabi.h](#)

## 5.61 `__gnu_cxx::rope<_CharT, _Alloc>` Class Template Reference

Inherits `__gnu_cxx::Rope_base<_CharT, _Alloc>`.

### Public Types

- `typedef _Rope_RopeConcatenation<_CharT, _Alloc> __C`
- `typedef _Rope_RopeFunction<_CharT, _Alloc> __F`
- `typedef _Rope_RopeLeaf<_CharT, _Alloc> __L`
- `typedef _Rope_RopeSubstring<_CharT, _Alloc> __S`
- `typedef _Alloc::template rebind<__C>::other _CAlloc`
- `typedef _Alloc::template rebind<_CharT>::other _DataAlloc`
- `typedef _Alloc::template rebind<__F>::other _FAlloc`
- `typedef _Alloc::template rebind<__L>::other _LAlloc`
- `typedef _Alloc::template rebind<__S>::other _SAlloc`
- `typedef _Rope_const_iterator<_CharT, _Alloc> const_iterator`
- `typedef const _CharT * const_pointer`
- `typedef _CharT const_reference`
- `typedef std::reverse\_iterator<const_iterator> const_reverse_iterator`
- `typedef ptrdiff_t difference_type`
- `typedef _Rope_iterator<_CharT, _Alloc> iterator`
- `typedef _Rope_char_ptr_proxy<_CharT, _Alloc> pointer`
- `typedef _Rope_char_ref_proxy<_CharT, _Alloc> reference`
- `typedef std::reverse\_iterator<iterator> reverse_iterator`
- `typedef size_t size_type`
- `typedef _CharT value_type`

### Public Member Functions

- **rope** (const `_CharT` \*`__s`, const `allocator_type` &`__a`=`allocator_type`())
- **rope** (const `_CharT` \*`__s`, `size_t` `__len`, const `allocator_type` &`__a`=`allocator_type`())
- **rope** (const `iterator` &`__s`, const `iterator` &`__e`, const `allocator_type` &`__a`=`allocator_type`())

- **rope** (`_CharT __c`, `const allocator_type &__a=allocator_type()`)
- **rope** (`size_t __n`, `_CharT __c`, `const allocator_type &__a=allocator_type()`)
- **rope** (`const allocator_type &__a=allocator_type()`)
- **rope** (`const _CharT *__s`, `const _CharT *__e`, `const allocator_type &__a=allocator_type()`)
- **rope** (`char_producer<_CharT> *__fn`, `size_t __len`, `bool __delete_fn`, `const allocator_type &__a=allocator_type()`)
- **rope** (`const rope &__x`, `const allocator_type &__a=allocator_type()`)
- **rope** (`const const_iterator &__s`, `const const_iterator &__e`, `const allocator_type &__a=allocator_type()`)
- `allocator_type &_M_get_allocator ()`
- `const allocator_type &_M_get_allocator () const`
- **rope & append** (`const _CharT *__iter`, `size_t __n`)
- **rope & append** (`const _CharT *__c_string`)
- **rope & append** (`const _CharT *__s`, `const _CharT *__e`)
- **rope & append** (`const_iterator __s`, `const_iterator __e`)
- **rope & append** (`_CharT __c`)
- **rope & append** ()
- **rope & append** (`const rope &__y`)
- **rope & append** (`size_t __n`, `_CharT __c`)
- **void apply\_to\_pieces** (`size_t __begin`, `size_t __end`, `_Rope_char_consumer<_CharT> &__c`) `const`
- `_CharT at` (`size_type __pos`) `const`
- `_CharT back` () `const`
- **void balance** ()
- `const_iterator begin` () `const`
- `const_iterator begin` ()
- `const _CharT * c_str` () `const`
- **void clear** ()
- `int compare` (`const rope &__y`) `const`
- `const_iterator const_begin` () `const`
- `const_iterator const_end` () `const`
- `const_reverse_iterator const_rbegin` () `const`
- `const_reverse_iterator const_rend` () `const`
- **void copy** (`_CharT *__buffer`) `const`
- `size_type copy` (`size_type __pos`, `size_type __n`, `_CharT *__buffer`) `const`
- **void delete\_c\_str** ()
- **void dump** ()
- `bool empty` () `const`
- `const_iterator end` () `const`
- `const_iterator end` ()
- **void erase** (`size_t __p`, `size_t __n`)
- **void erase** (`size_t __p`)

- iterator **erase** (const iterator &\_\_p, const iterator &\_\_q)
- iterator **erase** (const iterator &\_\_p)
- size\_type **find** (\_CharT \_\_c, size\_type \_\_pos=0) const
- size\_type **find** (const \_CharT \*\_\_s, size\_type \_\_pos=0) const
- \_CharT **front** () const
- allocator\_type **get\_allocator** () const
- iterator **insert** (const iterator &\_\_p, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- iterator **insert** (const iterator &\_\_p, const [rope](#) &\_\_r)
- iterator **insert** (const iterator &\_\_p, size\_t \_\_n, \_CharT \_\_c)
- iterator **insert** (const iterator &\_\_p, \_CharT \_\_c)
- iterator **insert** (const iterator &\_\_p)
- iterator **insert** (const iterator &\_\_p, const \_CharT \*c\_string)
- iterator **insert** (const iterator &\_\_p, const \_CharT \*\_\_i, size\_t \_\_n)
- iterator **insert** (const iterator &\_\_p, const const\_iterator &\_\_i, const const\_iterator &\_\_j)
- iterator **insert** (const iterator &\_\_p, const iterator &\_\_i, const iterator &\_\_j)
- void **insert** (size\_t \_\_p, const const\_iterator &\_\_i, const const\_iterator &\_\_j)
- void **insert** (size\_t \_\_p, size\_t \_\_n, \_CharT \_\_c)
- void **insert** (size\_t \_\_p, const \_CharT \*\_\_i, size\_t \_\_n)
- void **insert** (size\_t \_\_p)
- void **insert** (size\_t \_\_p, const [rope](#) &\_\_r)
- void **insert** (size\_t \_\_p, \_CharT \_\_c)
- void **insert** (size\_t \_\_p, const \_CharT \*\_\_c\_string)
- void **insert** (size\_t \_\_p, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- void **insert** (size\_t \_\_p, const iterator &\_\_i, const iterator &\_\_j)
- size\_type **length** () const
- size\_type **max\_size** () const
- iterator **mutable\_begin** ()
- iterator **mutable\_end** ()
- [reverse\\_iterator](#) **mutable\_rbegin** ()
- reference **mutable\_reference\_at** (size\_type \_\_pos)
- [reverse\\_iterator](#) **mutable\_rend** ()
- [rope](#) & **operator=** (const [rope](#) &\_\_x)
- \_CharT **operator[]** (size\_type \_\_pos) const
- void **pop\_back** ()
- void **pop\_front** ()
- void **push\_back** (\_CharT \_\_x)
- void **push\_front** (\_CharT \_\_x)
- [const\\_reverse\\_iterator](#) **rbegin** ()
- [const\\_reverse\\_iterator](#) **rbegin** () const
- [const\\_reverse\\_iterator](#) **rend** ()
- [const\\_reverse\\_iterator](#) **rend** () const
- void **replace** (size\_t \_\_p, const iterator &\_\_i, const iterator &\_\_j)

- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- void **replace** (size\_t \_\_p, const \_CharT \*\_\_i, size\_t \_\_i\_len)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, \_CharT \_\_c)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const \_CharT \*\_\_c\_string)
- void **replace** (const iterator &\_\_p, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- void **replace** (const iterator &\_\_p, iterator \_\_i, iterator \_\_j)
- void **replace** (const iterator &\_\_p, const [rope](#) &\_\_r)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const const\_iterator &\_\_i, const const\_iterator &\_\_j)
- void **replace** (size\_t \_\_p, size\_t \_\_n, const [rope](#) &\_\_r)
- void **replace** (size\_t \_\_p, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const [rope](#) &\_\_r)
- void **replace** (const iterator &\_\_p, \_CharT \_\_c)
- void **replace** (const iterator &\_\_p, const \_CharT \*\_\_i, size\_t \_\_n)
- void **replace** (size\_t \_\_p, size\_t \_\_n, const \_CharT \*\_\_c\_string)
- void **replace** (size\_t \_\_p, size\_t \_\_n, \_CharT \_\_c)
- void **replace** (size\_t \_\_p, size\_t \_\_n, const \_CharT \*\_\_i, size\_t \_\_i\_len)
- void **replace** (size\_t \_\_p, const [rope](#) &\_\_r)
- void **replace** (size\_t \_\_p, const \_CharT \*\_\_c\_string)
- void **replace** (size\_t \_\_p, size\_t \_\_n, const iterator &\_\_i, const iterator &\_\_j)
- void **replace** (size\_t \_\_p, size\_t \_\_n, const const\_iterator &\_\_i, const const\_iterator &\_\_j)
- void **replace** (size\_t \_\_p, \_CharT \_\_c)
- void **replace** (const iterator &\_\_p, const \_CharT \*\_\_c\_string)
- void **replace** (size\_t \_\_p, const const\_iterator &\_\_i, const const\_iterator &\_\_j)
- void **replace** (size\_t \_\_p, size\_t \_\_n, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const iterator &\_\_i, const iterator &\_\_j)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const \_CharT \*\_\_i, size\_t \_\_n)
- const \_CharT \* **replace\_with\_c\_str** ()
- size\_type **size** () const
- [rope](#)<\_CharT, \_Alloc> **substr** (const\_iterator \_\_start)
- [rope](#) **substr** (const\_iterator \_\_start, const\_iterator \_\_end) const
- [rope](#) **substr** (iterator \_\_start) const
- [rope](#) **substr** (size\_t \_\_start, size\_t \_\_len=1) const
- [rope](#) **substr** (iterator \_\_start, iterator \_\_end) const
- void **swap** ([rope](#) &\_\_b)

**Static Public Member Functions**

- static `__C * _C_allocate` (size\_t \_\_n)
- static void `_C_deallocate` (\_\_C \*\_\_p, size\_t \_\_n)
- static `_CharT * _Data_allocate` (size\_t \_\_n)
- static void `_Data_deallocate` (\_CharT \*\_\_p, size\_t \_\_n)
- static `__F * _F_allocate` (size\_t \_\_n)
- static void `_F_deallocate` (\_\_F \*\_\_p, size\_t \_\_n)
- static `__L * _L_allocate` (size\_t \_\_n)
- static void `_L_deallocate` (\_\_L \*\_\_p, size\_t \_\_n)
- static `__S * _S_allocate` (size\_t \_\_n)
- static void `_S_deallocate` (\_\_S \*\_\_p, size\_t \_\_n)

**Public Attributes**

- `_RopeRep * _M_tree_ptr`

**Static Public Attributes**

- static const size\_type `npos`

**Protected Types**

- enum { `_S_copy_max` }
- typedef `_Rope_base<_CharT, _Alloc>` **\_Base**
- typedef `_CharT * _Cstrptr`
- typedef `_Rope_RopeConcatenation<_CharT, _Alloc>` **\_RopeConcatenation**
- typedef `_Rope_RopeFunction<_CharT, _Alloc>` **\_RopeFunction**
- typedef `_Rope_RopeLeaf<_CharT, _Alloc>` **\_RopeLeaf**
- typedef `_Rope_RopeRep<_CharT, _Alloc>` **\_RopeRep**
- typedef `_Rope_RopeSubstring<_CharT, _Alloc>` **\_RopeSubstring**
- typedef `_Rope_self_destruct_ptr<_CharT, _Alloc>` **\_Self\_destruct\_ptr**
- typedef `_Base::allocator_type` **allocator\_type**

**Static Protected Member Functions**

- static size\_t `_S_allocated_capacity` (size\_t \_\_n)
- static bool `_S_apply_to_pieces` (\_Rope\_char\_consumer<\_CharT> &\_\_c, const \_RopeRep \*\_\_r, size\_t \_\_begin, size\_t \_\_end)
- static \_RopeRep \* `_S_concat` (\_RopeRep \*\_\_left, \_RopeRep \*\_\_right)
- static \_RopeRep \* `_S_concat_char_iter` (\_RopeRep \*\_\_r, const \_CharT \*\_\_iter, size\_t \_\_slen)

- static `_RopeRep * _S_destr_concat_char_iter` (`_RopeRep * __r`, const `_CharT * __iter`, size\_t `__slen`)
- static `_RopeLeaf * _S_destr_leaf_concat_char_iter` (`_RopeLeaf * __r`, const `_CharT * __iter`, size\_t `__slen`)
- static `_CharT _S_fetch` (`_RopeRep * __r`, size\_type `__pos`)
- static `_CharT * _S_fetch_ptr` (`_RopeRep * __r`, size\_type `__pos`)
- static bool `_S_is0` (`_CharT __c`)
- static `_RopeLeaf * _S_leaf_concat_char_iter` (`_RopeLeaf * __r`, const `_CharT * __iter`, size\_t `__slen`)
- static `_RopeConcatenation * _S_new_RopeConcatenation` (`_RopeRep * __left`, `_RopeRep * __right`, allocator\_type &`__a`)
- static `_RopeFunction * _S_new_RopeFunction` (char\_producer< `_CharT` > \*`__f`, size\_t `__size`, bool `__d`, allocator\_type &`__a`)
- static `_RopeLeaf * _S_new_RopeLeaf` (`_CharT * __s`, size\_t `__size`, allocator\_type &`__a`)
- static `_RopeSubstring * _S_new_RopeSubstring` (`_Rope_RopeRep< _CharT, _Alloc > * __b`, size\_t `__s`, size\_t `__l`, allocator\_type &`__a`)
- static void `_S_ref` (`_RopeRep * __t`)
- static `_RopeLeaf * _S_RopeLeaf_from_unowned_char_ptr` (const `_CharT * __s`, size\_t `__size`, allocator\_type &`__a`)
- static size\_t `_S_rounded_up_size` (size\_t `__n`)
- static `_RopeRep * _S_substring` (`_RopeRep * __base`, size\_t `__start`, size\_t `__endp1`)
- static `_RopeRep * _S_tree_concat` (`_RopeRep * __left`, `_RopeRep * __right`)
- static void `_S_unref` (`_RopeRep * __t`)
- static `_RopeRep * replace` (`_RopeRep * __old`, size\_t `__pos1`, size\_t `__pos2`, `_RopeRep * __r`)

#### Static Protected Attributes

- static `_CharT _S_empty_c_str` [1]

#### Friends

- class `_Rope_char_ptr_proxy< _CharT, _Alloc >`
- class `_Rope_char_ref_proxy< _CharT, _Alloc >`
- class `_Rope_const_iterator< _CharT, _Alloc >`
- class `_Rope_iterator< _CharT, _Alloc >`
- class `_Rope_iterator_base< _CharT, _Alloc >`
- struct `_Rope_RopeRep< _CharT, _Alloc >`
- struct `_Rope_RopeSubstring< _CharT, _Alloc >`



- `template<class _CharT2, class _Alloc2 >`  
`rope<_CharT2, _Alloc2 > operator+ (const rope<_CharT2, _Alloc2 > &__-`  
`left, const rope<_CharT2, _Alloc2 > &__right)`
- `template<class _CharT2, class _Alloc2 >`  
`rope<_CharT2, _Alloc2 > operator+ (const rope<_CharT2, _Alloc2 > &__-`  
`left, _CharT2 __right)`
- `template<class _CharT2, class _Alloc2 >`  
`rope<_CharT2, _Alloc2 > operator+ (const rope<_CharT2, _Alloc2 > &__-`  
`left, const _CharT2 *__right)`

### 5.61.1 Detailed Description

`template<class _CharT, class _Alloc> class __gnu_cxx::rope<_CharT, _Alloc >`

This is an SGI extension.

#### Todo

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

Definition at line 1510 of file `rope`.

The documentation for this class was generated from the following files:

- `rope`
- `ropeimpl.h`

## 5.62 `__gnu_cxx::select1st<_Pair>` Struct Template Reference

An [SGI extension](#) .

Inherits `std::_Select1st<_Pair>`.

### Public Types

- `typedef _Pair argument_type`
- `typedef _Pair::first_type result_type`

### Public Member Functions

- `_Pair::first_type & operator() (_Pair &__x) const`
- `template<typename _Pair2 >`  
`const _Pair2::first_type & operator() (const _Pair2 &__x) const`

- `template<typename _Pair2>`  
`_Pair2::first_type & operator() (_Pair2 &__x) const`
- `const _Pair::first_type & operator() (const _Pair &__x) const`

#### 5.62.1 Detailed Description

`template<class _Pair> struct __gnu_cxx::select1st<_Pair>`

An [SGI extension](#) .

Definition at line 200 of file `ext/functional`.

#### 5.62.2 Member Typedef Documentation

**5.62.2.1** `typedef _Pair std::unary_function<_Pair, _Pair::first_type>::argument_type [inherited]`

`argument_type` is the type of the argument

Definition at line 105 of file `stl_function.h`.

**5.62.2.2** `typedef _Pair::first_type std::unary_function<_Pair, _Pair::first_type>::result_type [inherited]`

`result_type` is the return type

Definition at line 108 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

### 5.63 `__gnu_cxx::select2nd<_Pair>` Struct Template Reference

An [SGI extension](#) .

Inherits `std::_Select2nd<_Pair>`.

#### Public Types

- `typedef _Pair` [argument\\_type](#)
- `typedef _Pair::second_type` [result\\_type](#)

### Public Member Functions

- `_Pair::second_type & operator() (_Pair &__x) const`
- `const _Pair::second_type & operator() (const _Pair &__x) const`

#### 5.63.1 Detailed Description

`template<class _Pair> struct __gnu_cxx::select2nd<_Pair>`

An [SGI extension](#) .

Definition at line 204 of file `ext/functional`.

#### 5.63.2 Member Typedef Documentation

**5.63.2.1** `typedef _Pair std::unary_function<_Pair, _Pair::second_type>::argument_type [inherited]`

`argument_type` is the type of the argument

Definition at line 105 of file `stl_function.h`.

**5.63.2.2** `typedef _Pair::second_type std::unary_function<_Pair, _Pair::second_type>::result_type [inherited]`

`result_type` is the return type

Definition at line 108 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

## 5.64 `__gnu_cxx::slist<_Tp, _Alloc>` Class Template Reference

Inherits `__gnu_cxx::Slist_base<_Tp, _Alloc>`.

### Public Types

- `typedef _Base::allocator_type allocator_type`
- `typedef _Slist_iterator<_Tp, const _Tp &, const _Tp * > const_iterator`

- `typedef const value_type * const_pointer`
- `typedef const value_type & const_reference`
- `typedef ptrdiff_t difference_type`
- `typedef _Slist_iterator<_Tp, _Tp &, _Tp * > iterator`
- `typedef value_type * pointer`
- `typedef value_type & reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

### Public Member Functions

- `slist` (const allocator\_type &\_\_a=allocator\_type())
- `slist` (size\_type \_\_n)
- `template<class _InputIterator >`  
`slist` (\_InputIterator \_\_first, \_InputIterator \_\_last, const allocator\_type &\_\_a=allocator\_type())
- `slist` (size\_type \_\_n, const value\_type &\_\_x, const allocator\_type &\_\_a=allocator\_type())
- `slist` (const `slist` &\_\_x)
- `template<class _Integer >`  
`void M_assign_dispatch` (\_Integer \_\_n, \_Integer \_\_val, \_\_true\_type)
- `template<class _InputIterator >`  
`void M_assign_dispatch` (\_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- `void M_fill_assign` (size\_type \_\_n, const \_Tp &\_\_val)
- `void assign` (size\_type \_\_n, const \_Tp &\_\_val)
- `template<class _InputIterator >`  
`void assign` (\_InputIterator \_\_first, \_InputIterator \_\_last)
- `iterator before_begin` ()
- `const_iterator before_begin` () const
- `iterator begin` ()
- `const_iterator begin` () const
- `void clear` ()
- `bool empty` () const
- `iterator end` ()
- `const_iterator end` () const
- `iterator erase` (iterator \_\_pos)
- `iterator erase` (iterator \_\_first, iterator \_\_last)
- `iterator erase_after` (iterator \_\_pos)
- `iterator erase_after` (iterator \_\_before\_first, iterator \_\_last)
- `reference front` ()
- `const_reference front` () const
- `allocator_type get_allocator` () const

- iterator **insert** (iterator \_\_pos, const value\_type &\_\_x)
- iterator **insert** (iterator \_\_pos)
- void **insert** (iterator \_\_pos, size\_type \_\_n, const value\_type &\_\_x)
- template<class \_InIterator >  
void **insert** (iterator \_\_pos, \_InIterator \_\_first, \_InIterator \_\_last)
- iterator **insert\_after** (iterator \_\_pos)
- void **insert\_after** (iterator \_\_pos, size\_type \_\_n, const value\_type &\_\_x)
- template<class \_InIterator >  
void **insert\_after** (iterator \_\_pos, \_InIterator \_\_first, \_InIterator \_\_last)
- iterator **insert\_after** (iterator \_\_pos, const value\_type &\_\_x)
- size\_type **max\_size** () const
- void **merge** (slist &\_\_x)
- template<class \_StrictWeakOrdering >  
void **merge** (slist &, \_StrictWeakOrdering)
- slist & **operator=** (const slist &\_\_x)
- void **pop\_front** ()
- iterator **previous** (const\_iterator \_\_pos)
- const\_iterator **previous** (const\_iterator \_\_pos) const
- void **push\_front** ()
- void **push\_front** (const value\_type &\_\_x)
- void **remove** (const \_Tp &\_\_val)
- template<class \_Predicate >  
void **remove\_if** (\_Predicate \_\_pred)
- void **resize** (size\_type new\_size, const \_Tp &\_\_x)
- void **resize** (size\_type new\_size)
- void **reverse** ()
- size\_type **size** () const
- void **sort** ()
- template<class \_StrictWeakOrdering >  
void **sort** (\_StrictWeakOrdering \_\_comp)
- void **splice** (iterator \_\_pos, slist &\_\_x, iterator \_\_i)
- void **splice** (iterator \_\_pos, slist &\_\_x)
- void **splice** (iterator \_\_pos, slist &\_\_x, iterator \_\_first, iterator \_\_last)
- void **splice\_after** (iterator \_\_pos, slist &\_\_x)
- void **splice\_after** (iterator \_\_pos, iterator \_\_prev)
- void **splice\_after** (iterator \_\_pos, iterator \_\_before\_first, iterator \_\_before\_last)
- void **swap** (slist &\_\_x)
- template<class \_BinaryPredicate >  
void **unique** (\_BinaryPredicate \_\_pred)
- void **unique** ()

### Private Types

- typedef `_Alloc::template rebind< _Slist_node< _Tp > >::other` **\_Node\_alloc**

**Private Member Functions**

- `_Slist_node_base * _M_erase_after (_Slist_node_base *__pos)`
- `_Slist_node_base * _M_erase_after (_Slist_node_base *, _Slist_node_base *)`
- `_Slist_node<_Tp> * _M_get_node ()`
- `void _M_put_node (_Slist_node<_Tp> *__p)`

**Private Attributes**

- `_Slist_node_base _M_head`

**5.64.1 Detailed Description**

`template<class _Tp, class _Alloc = allocator<_Tp>> class __gnu_cxx::slist<_Tp, _Alloc>`

This is an SGI extension.

**Todo**

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

Definition at line 293 of file `slist`.

The documentation for this class was generated from the following file:

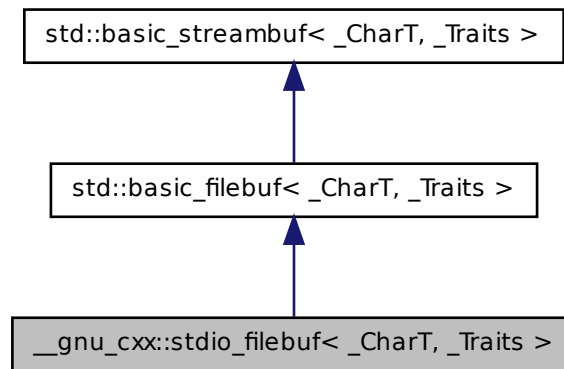
- [slist](#)

**5.65 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference**

Provides a layer of compatibility for C/POSIX.

This GNU extension provides extensions for working with standard C `FILE*`'s and POSIX file descriptors. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.

Inheritance diagram for `__gnu_cxx::stdio_filebuf<_CharT, _Traits>`:



### Public Types

- `typedef codecvt< char_type, char, __state_type > __codecvt_type`
- `typedef __basic_file< char > __file_type`
- `typedef basic_filebuf< char_type, traits_type > __filebuf_type`
- `typedef traits_type::state_type __state_type`
- `typedef basic_streambuf< char_type, traits_type > __streambuf_type`
- `typedef _CharT char_type`
- `typedef traits_type::int_type int_type`
- `typedef traits_type::off_type off_type`
- `typedef traits_type::pos_type pos_type`
- `typedef std::size_t size_t`
- `typedef _Traits traits_type`

### Public Member Functions

- `stdio_filebuf()`
- `stdio_filebuf(int __fd, std::ios_base::openmode __mode, size_t __size=static_cast< size_t >(BUFSIZ))`
- `stdio_filebuf(std::__c_file * __f, std::ios_base::openmode __mode, size_t __size=static_cast< size_t >(BUFSIZ))`

- virtual `~stdio_filebuf()`
- `__filebuf_type * close()`
- `int fd()`
- `std::__c_file * file()`
- `streamsize in_avail()`
- `bool is_open() const` throw `()`
- `__filebuf_type * open(const char *__s, ios_base::openmode __mode)`
- `__filebuf_type * open(const std::string &__s, ios_base::openmode __mode)`
- `int_type sbumpc()`
- `int_type sgetc()`
- `streamsize sgetn(char_type *__s, streamsize __n)`
- `int_type snextc()`
- `int_type sputbackc(char_type __c)`
- `int_type sputc(char_type __c)`
- `streamsize sputn(const char_type *__s, streamsize __n)`
- `int_type sungetc()`

#### Protected Member Functions

- `void _M_allocate_internal_buffer()`
- `bool _M_convert_to_external(char_type *, streamsize)`
- `void _M_create_pback()`
- `void _M_destroy_internal_buffer() throw()`
- `void _M_destroy_pback() throw()`
- `int _M_get_ext_pos(__state_type &__state)`
- `pos_type _M_seek(off_type __off, ios_base::seekdir __way, __state_type __state)`
- `void _M_set_buffer(streamsize __off)`
- `bool _M_terminate_output()`
- `void gbump(int __n)`
- `virtual void imbue(const locale &__loc)`
- `virtual int_type overflow(int_type __c=_Traits::eof())`
- `virtual int_type pbackfail(int_type __c=_Traits::eof())`
- `void pbump(int __n)`
- `virtual pos_type seekoff(off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)`
- `virtual pos_type seekpos(pos_type __pos, ios_base::openmode __mode=ios_base::in|ios_base::out)`
- `virtual __streambuf_type * setbuf(char_type *__s, streamsize __n)`
- `void setg(char_type *__gbeg, char_type *__gnext, char_type *__gend)`
- `void setp(char_type *__pbeg, char_type *__pend)`
- `virtual streamsize showmanyc()`
- `virtual int sync()`



- virtual `int_type` `uflow` ()
- virtual `int_type` `underflow` ()
- virtual `streamsize` `xsgetn` (`char_type` \* \_\_s, `streamsize` \_\_n)
- virtual `streamsize` `xspn` (const `char_type` \* \_\_s, `streamsize` \_\_n)
  
- `char_type` \* `eback` () const
- `char_type` \* `gptr` () const
- `char_type` \* `egptr` () const
  
- `char_type` \* `pbase` () const
- `char_type` \* `pptr` () const
- `char_type` \* `pptr` () const

### Protected Attributes

- `char_type` \* `_M_buf`
- bool `_M_buf_allocated`
- `size_t` `_M_buf_size`
- const `__codecvt_type` \* `_M_codecvt`
- `char` \* `_M_ext_buf`
- `streamsize` `_M_ext_buf_size`
- `char` \* `_M_ext_end`
- const `char` \* `_M_ext_next`
- `__file_type` `_M_file`
- `__c_lock` `_M_lock`
- `ios_base::openmode` `_M_mode`
- bool `_M_reading`
- `__state_type` `_M_state_beg`
- `__state_type` `_M_state_cur`
- `__state_type` `_M_state_last`
- bool `_M_writing`
  
- `char_type` `_M_pback`
- `char_type` \* `_M_pback_cur_save`
- `char_type` \* `_M_pback_end_save`
- bool `_M_pback_init`

### Friends

- `template<bool _IsMove, typename _CharT2 >`  
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, _CharT2 * >::__-`  
`type __copy_move_a2` (`istreambuf_iterator< _CharT2 >`, `istreambuf_iterator<`  
`_CharT2 >`, `_CharT2 *`)

- `streamsize __copy_streambufs_eof (__streambuf_type *, __streambuf_type *, bool &)`
  - class `basic_ios< char_type, traits_type >`
  - class `basic_istream< char_type, traits_type >`
  - class `basic_ostream< char_type, traits_type >`
  - `template<typename _CharT2 >`  
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, istreambuf_iterator< _CharT2 >::__type >::find (istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, const _CharT2 &)`
  - `template<typename _CharT2, typename _Traits2, typename _Alloc >`  
`basic_istream< _CharT2, _Traits2 > & getline (basic_istream< _CharT2, _Traits2 > &, basic_string< _CharT2, _Traits2, _Alloc > &, _CharT2)`
  - class `ios_base`
  - class `istreambuf_iterator< char_type, traits_type >`
  - `template<typename _CharT2, typename _Traits2 >`  
`basic_istream< _CharT2, _Traits2 > & operator>> (basic_istream< _CharT2, _Traits2 > &, _CharT2 *)`
  - `template<typename _CharT2, typename _Traits2, typename _Alloc >`  
`basic_istream< _CharT2, _Traits2 > & operator>> (basic_istream< _CharT2, _Traits2 > &, basic_string< _CharT2, _Traits2, _Alloc > &)`
  - class `ostreambuf_iterator< char_type, traits_type >`
- 
- locale `pubimbue` (const locale &\_\_loc)
  - locale `getloc` () const
  - `__streambuf_type * pubsetbuf (char_type *__s, streamsize __n)`
  - `pos_type pubseekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)`
  - `pos_type pubseekpos (pos_type __sp, ios_base::openmode __mode=ios_base::in|ios_base::out)`
  - int `pubsync` ()
  - `char_type * _M_in_beg`
  - `char_type * _M_in_cur`
  - `char_type * _M_in_end`
  - `char_type * _M_out_beg`
  - `char_type * _M_out_cur`
  - `char_type * _M_out_end`
  - locale `_M_buf_locale`

## 5.65 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference

### 5.65.1 Detailed Description

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
class __gnu_cxx::stdio_filebuf< _CharT, _Traits >
```

Provides a layer of compatibility for C/POSIX.

This GNU extension provides extensions for working with standard C FILE\*'s and POSIX file descriptors. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.

Definition at line 51 of file `stdio_filebuf.h`.

### 5.65.2 Member Typedef Documentation

**5.65.2.1** `template<typename _CharT, typename _Traits> typedef basic_streambuf<char_type, traits_type> std::basic_filebuf<_CharT, _Traits>::__streambuf_type [inherited]`

This is a non-standard type.

Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits>](#).

Definition at line 79 of file `fstream`.

**5.65.2.2** `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> typedef _CharT __gnu_cxx::stdio_filebuf<_CharT, _Traits>::__char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_filebuf<\\_CharT, \\_Traits>](#).

Definition at line 55 of file `stdio_filebuf.h`.

**5.65.2.3** `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> typedef traits_type::int_type __gnu_cxx::stdio_filebuf<_CharT, _Traits>::__int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_filebuf<\\_CharT, \\_Traits>](#).

## 5.65 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 1001

Definition at line 57 of file `stdio_filebuf.h`.

**5.65.2.4** `template<typename _CharT, typename _Traits =  
std::char_traits<_CharT>> typedef traits_type::off_type  
__gnu_cxx::stdio_filebuf<_CharT, _Traits>::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_filebuf<_CharT, _Traits>`.

Definition at line 59 of file `stdio_filebuf.h`.

**5.65.2.5** `template<typename _CharT, typename _Traits =  
std::char_traits<_CharT>> typedef traits_type::pos_type  
__gnu_cxx::stdio_filebuf<_CharT, _Traits>::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_filebuf<_CharT, _Traits>`.

Definition at line 58 of file `stdio_filebuf.h`.

**5.65.2.6** `template<typename _CharT, typename _Traits = std::char_traits<_  
_CharT>> typedef _Traits __gnu_cxx::stdio_filebuf<_CharT, _Traits  
>::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_filebuf<_CharT, _Traits>`.

Definition at line 56 of file `stdio_filebuf.h`.

### 5.65.3 Constructor & Destructor Documentation

**5.65.3.1** `template<typename _CharT, typename _Traits =  
std::char_traits<_CharT>> __gnu_cxx::stdio_filebuf<_CharT,  
_Traits>::stdio_filebuf( ) [inline]`

deferred initialization

Definition at line 66 of file `stdio_filebuf.h`.

**5.65.3.2** `template<typename _CharT, typename _Traits>  
 __gnu_cxx::stdio_filebuf<_CharT, _Traits>::stdio_filebuf  
 ( int __fd, std::ios_base::openmode __mode, size_t __size =  
 static_cast<size_t>(BUFSIZ) )`

#### Parameters

- fd* An open file descriptor.
- mode* Same meaning as in a standard filebuf.
- size* Optimal or preferred size of internal buffer, in chars.

This constructor associates a file stream buffer with an open POSIX file descriptor. The file descriptor will be automatically closed when the `stdio_filebuf` is closed/destroyed.

Definition at line 128 of file `stdio_filebuf.h`.

References `std::basic_filebuf<_CharT, _Traits>::_M_buf_size`, `std::basic_filebuf<_CharT, _Traits>::_M_mode`, `std::basic_filebuf<_CharT, _Traits>::_M_reading`, `std::basic_filebuf<_CharT, _Traits>::_M_set_buffer()`, and `std::basic_filebuf<_CharT, _Traits>::is_open()`.

**5.65.3.3** `template<typename _CharT, typename _Traits>  
 __gnu_cxx::stdio_filebuf<_CharT, _Traits>::stdio_filebuf (   
 std::_c_file * __f, std::ios_base::openmode __mode, size_t __size =  
 static_cast<size_t>(BUFSIZ) )`

#### Parameters

- f* An open `FILE*`.
- mode* Same meaning as in a standard filebuf.
- size* Optimal or preferred size of internal buffer, in chars. Defaults to system's `BUFSIZ`.

This constructor associates a file stream buffer with an open C `FILE*`. The `FILE*` will not be automatically closed when the `stdio_filebuf` is closed/destroyed.

Definition at line 144 of file `stdio_filebuf.h`.

References `std::basic_filebuf<_CharT, _Traits>::_M_buf_size`, `std::basic_filebuf<_CharT, _Traits>::_M_mode`, `std::basic_filebuf<_CharT, _Traits>::_M_reading`, `std::basic_filebuf<_CharT, _Traits>::_M_set_buffer()`, and `std::basic_filebuf<_CharT, _Traits>::is_open()`.

**5.65.3.4** `template<typename _CharT, typename _Traits>`  
`__gnu_cxx::stdio_filebuf<_CharT, _Traits>::~~stdio_filebuf ( )`  
`[virtual]`

Closes the external data stream if the file descriptor constructor was used.

Definition at line 123 of file `stdio_filebuf.h`.

#### 5.65.4 Member Function Documentation

**5.65.4.1** `template<typename _CharT, typename _Traits> void`  
`std::basic_filebuf<_CharT, _Traits>::_M_create_pback ( )`  
`[inline, protected, inherited]`

Initializes pback buffers, and moves normal buffers to safety. Assumptions: `_M_in_cur` has already been moved back

Definition at line 174 of file `fstream`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::pbackfail()`.

**5.65.4.2** `template<typename _CharT, typename _Traits> void`  
`std::basic_filebuf<_CharT, _Traits>::_M_destroy_pback ( ) throw`  
`() [inline, protected, inherited]`

Deactivates pback buffer contents, and restores normal buffer. Assumptions: The pback buffer has only moved forward.

Definition at line 191 of file `fstream`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::seekoff()`, `std::basic_filebuf<_CharT, _Traits>::seekpos()`, `std::basic_filebuf<_CharT, _Traits>::underflow()`, and `std::basic_filebuf<_CharT, _Traits>::xsgetn()`.

**5.65.4.3** `template<typename _CharT, typename _Traits> void`  
`std::basic_filebuf<_CharT, _Traits>::_M_set_buffer ( streamsize`  
`__off ) [inline, protected, inherited]`

This function sets the pointers of the internal buffer, both get and put areas. Typically: `__off == egptr() - eback()` upon underflow/uflow (**read** mode); `__off == 0` upon overflow (**write** mode); `__off == -1` upon open, setbuf, seekoff/pos (**uncommitted** mode).

NB: `epptr() - pbase() == _M_buf_size - 1`, since `_M_buf_size` reflects the actual allocated memory and the last cell is reserved for the overflow char of a full put area.

Definition at line 392 of file `fstream`.

## 5.65 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 1004

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, `std::basic_filebuf<_CharT, _Traits>::open()`, `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::pbackfail()`, `__gnu_cxx::stdio_filebuf<_CharT, _Traits>::stdio_filebuf()`, `std::basic_filebuf<_CharT, _Traits>::underflow()`, `std::basic_filebuf<_CharT, _Traits>::xsgetn()`, and `std::basic_filebuf<_CharT, _Traits>::xsputn()`.

### 5.65.4.4 `template<typename _CharT, typename _Traits> basic_filebuf<_CharT, _Traits>::_filebuf_type * std::basic_filebuf<_CharT, _Traits>::close ( ) [inherited]`

Closes the currently associated file.

#### Returns

`this` on success, `NULL` on failure

If no file is currently open, this function immediately fails.

If a *put buffer area* exists, `overflow(eof)` is called to flush all the characters. The file is then closed.

If any operations fail, this function also fails.

Definition at line 130 of file `fstream.tcc`.

References `std::basic_filebuf<_CharT, _Traits>::is_open()`.

Referenced by `std::basic_fstream<_CharT, _Traits>::close()`, `std::basic_ofstream<_CharT, _Traits>::close()`, `std::basic_ifstream<_CharT, _Traits>::close()`, `std::basic_filebuf<_CharT, _Traits>::open()`, and `std::basic_filebuf<char_type, traits_type>::~~basic_filebuf()`.

### 5.65.4.5 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::eback ( ) const [inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence

- `egptr()` returns the end pointer for the input sequence

Definition at line 462 of file `streambuf`.

Referenced by `std::basic_filebuf<char_type, traits_type>::_M_destroy_pback()`, `std::basic_filebuf<_CharT, _Traits>::_imbue()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::pbackfail()`, `std::basic_filebuf<_CharT, _Traits>::pbackfail()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, `std::basic_streambuf<char_type, traits_type>::sputbackc()`, `std::basic_streambuf<char_type, traits_type>::sungetc()`, `std::basic_filebuf<_CharT, _Traits>::underflow()`, and `std::basic_filebuf<_CharT, _Traits>::xsgetn()`.

**5.65.4.6** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::egptr ( ) const  
[inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 468 of file `streambuf`.

Referenced by `std::basic_filebuf<char_type, traits_type>::_M_create_pback()`, `std::basic_streambuf<char_type, traits_type>::in_avail()`, `std::basic_streambuf<char_type, traits_type>::sbumpc()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, `std::basic_streambuf<char_type, traits_type>::sgetc()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::showmanyc()`, `std::basic_filebuf<_CharT, _Traits>::showmanyc()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::str()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::underflow()`, `std::basic_filebuf<_CharT, _Traits>::underflow()`, `std::basic_streambuf<_CharT, _Traits>::xsgetn()`, and `std::basic_filebuf<_CharT, _Traits>::xsgetn()`.

**5.65.4.7** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::eptr ( ) const  
[inline, protected, inherited]`



## 5.65 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 1006

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 515 of file `streambuf`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_streambuf<char_type, traits_type>::sputc()`, `std::basic_streambuf<_CharT, _Traits>::xsputn()`, and `std::basic_filebuf<_CharT, _Traits>::xsputn()`.

**5.65.4.8** `template<typename _CharT, typename _Traits =  
std::char_traits<_CharT>> int __gnu_cxx::stdio_filebuf<_CharT,  
_Traits>::fd ( ) [inline]`

### Returns

The underlying file descriptor.

Once associated with an external data stream, this function can be used to access the underlying POSIX file descriptor. Note that there is no way for the library to track what you do with the descriptor, so be careful.

Definition at line 109 of file `stdio_filebuf.h`.

**5.65.4.9** `template<typename _CharT, typename _Traits =  
std::char_traits<_CharT>> std::_c_file* __gnu_cxx::stdio_filebuf<  
_CharT, _Traits>::file ( ) [inline]`

### Returns

The underlying `FILE*`.

This function can be used to access the underlying "C" file pointer. Note that there is no way for the library to track what you do with the file, so be careful.

Definition at line 119 of file `stdio_filebuf.h`.

**5.65.4.10** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf<_CharT, _Traits>::gbump ( int __n )  
[inline, protected, inherited]`

Moving the read position.

#### Parameters

*n* The delta by which to move.

This just advances the read position without returning any data.

Definition at line 478 of file `streambuf`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::pbackfail()`, `std::basic_filebuf<_CharT, _Traits>::pbackfail()`, `std::basic_streambuf<char_type, traits_type>::sbumpc()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, `std::basic_streambuf<char_type, traits_type>::sputbackc()`, `std::basic_streambuf<char_type, traits_type>::sungetc()`, `std::basic_streambuf<char_type, traits_type>::uflow()`, `std::basic_streambuf<_CharT, _Traits>::xsgetn()`, and `std::basic_filebuf<_CharT, _Traits>::xsgetn()`.

**5.65.4.11** `template<typename _CharT, typename _Traits> locale  
std::basic_streambuf<_CharT, _Traits>::getloc ( ) const  
[inline, inherited]`

Locale access.

#### Returns

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 224 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::pubimbue()`.

**5.65.4.12** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::gptr ( ) const  
[inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 465 of file `streambuf`.

Referenced by `std::basic_filebuf<char_type, traits_type>::_M_create_pback()`, `std::basic_filebuf<char_type, traits_type>::_M_destroy_pback()`, `std::basic_filebuf<_CharT, _Traits>::imbue()`, `std::basic_streambuf<char_type, traits_type>::in_avail()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::pbackfail()`, `std::basic_filebuf<_CharT, _Traits>::pbackfail()`, `std::basic_streambuf<char_type, traits_type>::sbumpc()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, `std::basic_streambuf<char_type, traits_type>::sgetc()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::showmanyc()`, `std::basic_filebuf<_CharT, _Traits>::showmanyc()`, `std::basic_streambuf<char_type, traits_type>::sputbackc()`, `std::basic_streambuf<char_type, traits_type>::sungetc()`, `std::basic_streambuf<char_type, traits_type>::uflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::underflow()`, `std::basic_filebuf<_CharT, _Traits>::underflow()`, `std::basic_streambuf<_CharT, _Traits>::xsgetn()`, and `std::basic_filebuf<_CharT, _Traits>::xsgetn()`.

**5.65.4.13** `template<typename _CharT, typename _Traits> void  
std::basic_filebuf<_CharT, _Traits>::imbue ( const locale & )  
[protected, virtual, inherited]`

Changes translations.

#### Parameters

*loc* A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

#### Note

Base class version does nothing.

## 5.65 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 914 of file `fstream.tcc`.

References `std::basic_filebuf<_CharT, _Traits>::_M_ext_buf`, `std::basic_filebuf<_CharT, _Traits>::_M_ext_next`, `std::basic_filebuf<_CharT, _Traits>::_M_mode`, `std::basic_filebuf<_CharT, _Traits>::_M_reading`, `std::basic_filebuf<_CharT, _Traits>::_M_set_buffer()`, `std::ios_base::cur`, `std::basic_streambuf<_CharT, _Traits>::eback()`, `std::basic_streambuf<_CharT, _Traits>::gptr()`, `std::basic_filebuf<_CharT, _Traits>::is_open()`, and `std::basic_filebuf<_CharT, _Traits>::seekoff()`.

### 5.65.4.14 `template<typename _CharT, typename _Traits> streamsize std::basic_streambuf<_CharT, _Traits>::in_avail ( ) [inline, inherited]`

Looking ahead into the stream.

#### Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

Definition at line 264 of file `streambuf`.

### 5.65.4.15 `template<typename _CharT, typename _Traits> bool std::basic_filebuf<_CharT, _Traits>::is_open ( ) const throw () [inline, inherited]`

Returns true if the external file is open.

Definition at line 224 of file `fstream`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::close()`, `std::basic_filebuf<_CharT, _Traits>::imbue()`, `std::basic_fstream<_CharT, _Traits>::is_open()`, `std::basic_ofstream<_CharT, _Traits>::is_open()`, `std::basic_ifstream<_CharT, _Traits>::is_open()`, `std::basic_filebuf<_CharT, _Traits>::open()`, `std::basic_filebuf<_CharT, _Traits>::seekoff()`, `std::basic_filebuf<_CharT, _Traits>::seekpos()`, `std::basic_filebuf<_CharT, _Traits>::setbuf()`, `std::basic_filebuf<_CharT, _Traits>::showmanyc()`, and `__gnu_cxx::stdio_filebuf<_CharT, _Traits>::stdio_filebuf()`.

```

_CharT, _Traits >::__filebuf_type * std::basic_filebuf< _CharT,
_Traits >::open (const char * __s, ios_base::openmode __mode)
[inherited]

```

Opens an external file.

## Parameters

*s* The name of the file.

*mode* The open mode flags.

## Returns

this on success, NULL on failure

If a file is already open, this function immediately fails. Otherwise it tries to open the file named *s* using the flags given in *mode*.

Table 92, adapted here, gives the relation between openmode combinations and the equivalent fopen() flags. (NB: lines app, in/out|app, in|app, binary|app, binary|in/out|app, and binary|in|app per DR 596) +-----

```

-----+ | ios_base Flag combination stdio equivalent | |binary in out trunc app
| +-----+ | + w | | + a | | + a | | + w | | +
r | | + r+ | | + + w+ | | + + a+ | | + + a+ | +-----
-----+ | + + wb | | + + + ab | | + + ab | | + + + wb | | + + rb | | + + + r+b | | + + + +
w+b | | + + + + a+b | | + + + a+b | +-----+

```

Definition at line 96 of file fstream.tcc.

References `std::basic_filebuf<_CharT, _Traits >::M_mode`, `std::basic_filebuf<_CharT, _Traits >::M_reading`, `std::basic_filebuf<_CharT, _Traits >::M_set_buffer()`, `std::ios_base::ate`, `std::basic_filebuf<_CharT, _Traits >::close()`, `std::ios_base::end`, `std::basic_filebuf<_CharT, _Traits >::is_open()`, and `std::basic_filebuf<_CharT, _Traits >::seekoff()`.

Referenced by `std::basic_fstream<_CharT, _Traits >::open()`, `std::basic_ofstream<_CharT, _Traits >::open()`, and `std::basic_ifstream<_CharT, _Traits >::open()`.

```
template<typename _CharT, typename _Traits> __filebuf_type*
std::basic_filebuf< _CharT, _Traits >::open (const std::string &
__s, ios_base::openmode __mode) [inline, inherited]
```

Opens an external file.

## 5.65 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 1011

### Parameters

- s* The name of the file.
- mode* The open mode flags.

### Returns

`this` on success, `NULL` on failure

Definition at line 277 of file `fstream`.

Referenced by `std::basic_filebuf<char_type, traits_type>::open()`.

**5.65.4.18** `template<typename _CharT, typename _Traits> basic_filebuf<_CharT, _Traits>::int_type std::basic_filebuf<_CharT, _Traits>::overflow ( int_type = _Traits::eof() ) [protected, virtual, inherited]`

Consumes data from the buffer; writes to the controlled sequence.

### Parameters

- c* An additional character to consume.

### Returns

`eof()` to indicate failure, something else (usually *c*, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if *c* is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

### Note

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 424 of file `fstream.tcc`.

## 5.65 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference

References `std::basic_filebuf<_CharT, _Traits>::_M_buf_size`, `std::basic_filebuf<_CharT, _Traits>::_M_destroy_pback()`, `std::basic_filebuf<_CharT, _Traits>::_M_mode`, `std::basic_filebuf<_CharT, _Traits>::_M_reading`, `std::basic_filebuf<_CharT, _Traits>::_M_set_buffer()`, `std::ios_base::cur`, `std::ios_base::out`, `std::basic_streambuf<_CharT, _Traits>::pbase()`, `std::basic_streambuf<_CharT, _Traits>::pbump()`, and `std::basic_streambuf<_CharT, _Traits>::pptr()`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::pbackfail()`, `std::basic_filebuf<_CharT, _Traits>::sync()`, `std::basic_filebuf<_CharT, _Traits>::underflow()`, and `std::basic_filebuf<_CharT, _Traits>::xsgetn()`.

**5.65.4.19** `template<typename _CharT, typename _Traits> basic_filebuf<_CharT, _Traits>::int_type std::basic_filebuf<_CharT, _Traits>::pbackfail( int_type = _Traits::eof() )` [protected, virtual, inherited]

Tries to back up the input sequence.

### Parameters

*c* The character to be inserted back into the sequence.

### Returns

`eof()` on failure, *some other value* on success

### Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

### Note

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 365 of file `fstream.tcc`.

References `std::basic_filebuf<_CharT, _Traits>::_M_create_pback()`, `std::basic_filebuf<_CharT, _Traits>::_M_mode`, `std::basic_filebuf<_CharT, _Traits>::_M_pback_init`, `std::basic_filebuf<_CharT, _Traits>::_M_reading`, `std::basic_filebuf<_CharT, _Traits>::_M_set_buffer()`, `std::ios_base::cur`, `std::basic_streambuf<_CharT, _Traits>::eback()`, `std::basic_streambuf<_CharT, _Traits>::gbump()`, `std::basic_streambuf<_CharT, _Traits>::gptr()`, `std::ios_base::in`, `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::seekoff()`, and `std::basic_filebuf<_CharT, _Traits>::underflow()`.

**5.65.4.20** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::pbase ( ) const  
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 509 of file `streambuf`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_filebuf<_CharT, _Traits>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::str()`, `std::basic_filebuf<_CharT, _Traits>::sync()`, and `std::basic_filebuf<_CharT, _Traits>::xsputn()`.

**5.65.4.21** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf<_CharT, _Traits>::pbump ( int __n )  
[inline, protected, inherited]`

Moving the write position.

#### Parameters

- `n` The delta by which to move.

This just advances the write position without returning any data.

Definition at line 525 of file `streambuf`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, `std::basic_streambuf<char_type, traits_type>::sputc()`, and `std::basic_streambuf<_CharT, _Traits>::xsputn()`.



**5.65.4.22** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::pptr ( ) const  
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 512 of file `streambuf`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_filebuf<_CharT, _Traits>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, `std::basic_streambuf<char_type, traits_type>::sputc()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::str()`, `std::basic_filebuf<_CharT, _Traits>::sync()`, `std::basic_streambuf<_CharT, _Traits>::xsputn()`, and `std::basic_filebuf<_CharT, _Traits>::xsputn()`.

**5.65.4.23** `template<typename _CharT, typename _Traits> locale  
std::basic_streambuf<_CharT, _Traits>::pubimbue ( const locale  
& __loc ) [inline, inherited]`

Entry point for `imbue()`.

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 207 of file `streambuf`.

**5.65.4.24** `template<typename _CharT, typename _Traits> pos_type  
std::basic_streambuf<_CharT, _Traits>::pubseekoff ( off_type  
__off, ios_base::seekdir __way, ios_base::openmode __mode =  
ios_base::in | ios_base::out ) [inline, inherited]`

Entry point for `imbue()`.

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 241 of file `streambuf`.

**5.65.4.25** `template<typename _CharT, typename _Traits> pos_type  
std::basic_streambuf<_CharT, _Traits>::pubseekpos ( pos_type  
__sp, ios_base::openmode __mode = ios_base::in | ios_base::out )  
[inline, inherited]`

Entry point for `imbue()`.

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 246 of file `streambuf`.

**5.65.4.26** `template<typename _CharT, typename _Traits> __streambuf_type*  
std::basic_streambuf<_CharT, _Traits>::pubsetbuf ( char_type *  
__s, streamsize __n ) [inline, inherited]`

Entry points for derived buffer functions.

## 5.65 \_\_gnu\_cxx::stdio\_filebuf<\_CharT, \_Traits> Class Template Reference 1016

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 237 of file `streambuf`.

**5.65.4.27** `template<typename _CharT, typename _Traits> int  
std::basic_streambuf<_CharT, _Traits>::pubsync( ) [inline,  
inherited]`

Entry point for `imbue()`.

### Parameters

*loc* The new locale.

### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 251 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::sync()`.

**5.65.4.28** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf<_CharT, _Traits>::sbumpc( ) [inline,  
inherited]`

Getting the next character.

### Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 296 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::istreambuf_iterator<_CharT, _Traits>::operator++()`, and `std::basic_streambuf<char_type, traits_type>::snextc()`.

**5.65.4.29** `template<typename _CharT, typename _Traits> basic_filebuf<_CharT, _Traits>::pos_type std::basic_filebuf<_CharT, _Traits>::seekoff ( off_type, ios_base::seekdir, ios_base::openmode = ios_base::in | ios_base::out ) [protected, virtual, inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

#### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 717 of file `fstream.tcc`.

References `std::basic_filebuf<_CharT, _Traits>::_M_destroy_pback()`, `std::basic_filebuf<_CharT, _Traits>::_M_reading`, `std::ios_base::cur`, `std::basic_filebuf<_CharT, _Traits>::is_open()`, `std::basic_streambuf<_CharT, _Traits>::pbase()`, and `std::basic_streambuf<_CharT, _Traits>::pptr()`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, `std::basic_filebuf<_CharT, _Traits>::open()`, and `std::basic_filebuf<_CharT, _Traits>::pbackfail()`.

**5.65.4.30** `template<typename _CharT, typename _Traits> basic_filebuf<_CharT, _Traits>::pos_type std::basic_filebuf<_CharT, _Traits>::seekpos ( pos_type, ios_base::openmode = ios_base::in | ios_base::out ) [protected, virtual, inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

#### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 777 of file `fstream.tcc`.

References `std::basic_filebuf<_CharT, _Traits>::_M_destroy_pback()`, `std::ios_base::beg`, and `std::basic_filebuf<_CharT, _Traits>::is_open()`.

**5.65.4.31** `template<typename _CharT, typename _Traits> basic_filebuf<_CharT, _Traits>::__streambuf_type * std::basic_filebuf<_CharT, _Traits>::setbuf ( char_type * __s, streamsize __n )` `[protected, virtual, inherited]`

Manipulates the buffer.

#### Parameters

- s* Pointer to a buffer area.
- n* Size of *s*.

#### Returns

`this`

If no file has been opened, and both *s* and *n* are zero, then the stream becomes unbuffered. Otherwise, *s* is used as a buffer; see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 688 of file `fstream.tcc`.

References `std::basic_filebuf<_CharT, _Traits>::__M_buf`, `std::basic_filebuf<_CharT, _Traits>::__M_buf_size`, and `std::basic_filebuf<_CharT, _Traits>::is_open()`.

**5.65.4.32** `template<typename _CharT, typename _Traits> void std::basic_streambuf<_CharT, _Traits>::setg ( char_type * __gbeg, char_type * __gnext, char_type * __gend )` `[inline, protected, inherited]`

Setting the three read area pointers.

#### Parameters

- gbeg* A pointer.
- gnext* A pointer.
- gend* A pointer.

#### Postcondition

*gbeg* == `eback()`, *gnext* == `gptr()`, and *gend* == `egptr()`

## 5.65 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 1019

Definition at line 489 of file `streambuf`.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_create_pback()`, `std::basic_filebuf< char_type, traits_type >::_M_destroy_pback()`, and `std::basic_filebuf< char_type, traits_type >::_M_set_buffer()`.

**5.65.4.33** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf< _CharT, _Traits >::setp ( char_type *  
__pbeg, char_type * __pend ) [inline, protected,  
inherited]`

Setting the three write area pointers.

### Parameters

*pbeg* A pointer.

*pend* A pointer.

### Postcondition

*pbeg* == `pbase()`, *pbeg* == `pptr()`, and *pend* == `epptr()`

Definition at line 535 of file `streambuf`.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_set_buffer()`.

**5.65.4.34** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf< _CharT, _Traits >::sgetc ( ) [inline,  
inherited]`

Getting the next character.

### Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 318 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_streambuf< char_type, traits_type >::snextc()`.

**5.65.4.35** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf<_CharT, _Traits>::sgetn ( char_type * __s,  
streamsize __n ) [inline, inherited]`

Entry point for `xsgetn`.

#### Parameters

*s* A buffer area.

*n* A count.

Returns `xsgetn(s,n)`. The effect is to fill `s[0]` through `s[n-1]` with characters from the input sequence, if possible.

Definition at line 337 of file `streambuf`.

**5.65.4.36** `template<typename _CharT, typename _Traits> streamsize  
std::basic_filebuf<_CharT, _Traits>::showmanyc ( )  
[protected, virtual, inherited]`

Investigating the data available.

#### Returns

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.*  
[27.5.2.4.3]/1

#### Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 180 of file `fstream.tcc`.

## 5.65 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference

References `std::basic_filebuf<_CharT, _Traits>::_M_mode`, `std::ios_base::binary`, `std::basic_streambuf<_CharT, _Traits>::egptr()`, `std::basic_streambuf<_CharT, _Traits>::gptr()`, `std::ios_base::in`, and `std::basic_filebuf<_CharT, _Traits>::is_open()`.

**5.65.4.37** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf<_CharT, _Traits>::snextc ( ) [inline,  
inherited]`

Getting the next character.

### Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 278 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

**5.65.4.38** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf<_CharT, _Traits>::sputbackc ( char_type  
_c ) [inline, inherited]`

Pushing characters back into the input stream.

### Parameters

*c* The character to push back.

### Returns

The previous character, if possible.

Similar to `sungetc()`, but *c* is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be *c*.

Definition at line 352 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::putback()`.



**5.65.4.39** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf<_CharT, _Traits>::sputc ( char_type __c )  
[inline, inherited]`

Entry point for all single-character output functions.

#### Parameters

*c* A character to output.

#### Returns

*c*, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores *c* in that position, increments the position, and returns `traits::to_int_type(c)`. If a write position is not available, returns `overflow(c)`.

Definition at line 404 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, and `std::ostreambuf_iterator<_CharT, _Traits>::operator=()`.

**5.65.4.40** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf<_CharT, _Traits>::sputn ( const char_type *  
__s, streamsize __n ) [inline, inherited]`

Entry point for all single-character output functions.

#### Parameters

*s* A buffer read area.

*n* A count.

One of two public output functions.

Returns `xputn(s,n)`. The effect is to write `s[0]` through `s[n-1]` to the output sequence, if possible.

Definition at line 430 of file `streambuf`.

**5.65.4.41** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf<_CharT, _Traits>::sungetc ( ) [inline,  
inherited]`

Moving backwards in the input stream.

#### Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbackfail()`. The effect is to *unget* the last character *gotten*.

Definition at line 377 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::unget()`.

**5.65.4.42** `template<typename _CharT, typename _Traits> int  
std::basic_filebuf<_CharT, _Traits>::sync ( ) [protected,  
virtual, inherited]`

Synchronizes the buffer arrays with the controlled sequences.

#### Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

#### Note

Base class version does nothing, returns zero.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 897 of file `fstream.tcc`.

References `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_streambuf<_CharT, _Traits>::pbase()`, and `std::basic_streambuf<_CharT, _Traits>::pptr()`.

**5.65.4.43** `template<typename _CharT, typename _Traits> virtual int_type  
std::basic_streambuf<_CharT, _Traits>::uflow( ) [inline,  
protected, virtual, inherited]`

Fetches more data from the controlled sequence.

#### Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`.

Definition at line 680 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::sbumpc()`, and `std::basic_streambuf<_CharT, _Traits>::xsgetn()`.

**5.65.4.44** `template<typename _CharT, typename _Traits> basic_filebuf<  
_CharT, _Traits>::int_type std::basic_filebuf<_CharT, _Traits  
>::underflow( ) [protected, virtual, inherited]`

Fetches more data from the controlled sequence.

#### Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25.html>

#### Note

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 206 of file `fstream.tcc`.

References `std::basic_filebuf<_CharT, _Traits>::_M_buf_size`, `std::basic_filebuf<_CharT, _Traits>::_M_destroy_pback()`, `std::basic_filebuf<_CharT, _Traits>::_M_ext_buf`, `std::basic_filebuf<_CharT, _Traits>::_M_ext_buf_size`, `std::basic_filebuf<_CharT, _Traits>::_M_ext_next`, `std::basic_filebuf<_CharT, _Traits>::_M_mode`, `std::basic_filebuf<_CharT, _Traits>::_M_reading`, `std::basic_filebuf<_CharT, _Traits>::_M_set_buffer()`, `std::basic_streambuf<_CharT, _Traits>::eback()`, `std::basic_streambuf<_CharT, _Traits>::egptr()`, `std::basic_streambuf<_CharT, _Traits>::gptr()`, `std::__codecvt_abstract_base<_InternT, _ExternT, _StateT>::in()`, `std::ios_base::in`, `std::min()`, and `std::basic_filebuf<_CharT, _Traits>::overflow()`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::pbackfail()`.

**5.65.4.45** `template<typename _CharT, typename _Traits> streamsize  
std::basic_filebuf<_CharT, _Traits>::xsgetn ( char_type * __s,  
streamsize __n ) [protected, virtual, inherited]`

Multiple character extraction.

#### Parameters

- s* A buffer area.
- n* Maximum number of characters to assign.

#### Returns

The number of characters assigned.

Fills `s[0]` through `s[n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either *n* characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 551 of file `fstream.tcc`.

References `std::basic_filebuf<_CharT, _Traits>::_M_buf_size`, `std::basic_filebuf<_CharT, _Traits>::_M_destroy_pback()`, `std::basic_filebuf<_CharT, _Traits>::_M_mode`, `std::basic_filebuf<_CharT, _Traits>::_M_pback_init`, `std::basic_filebuf<_CharT, _Traits>::_M_reading`, `std::basic_filebuf<_CharT, _Traits>::_M_set_buffer()`, `std::basic_streambuf<_CharT, _Traits>::eback()`, `std::basic_streambuf<_CharT, _Traits>::egptr()`, `std::basic_streambuf<_CharT, _Traits>::gbump()`,

## 5.65 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 1026

`std::basic_streambuf<_CharT, _Traits>::gptr()`, `std::ios_base::in`, `std::basic_filebuf<_CharT, _Traits>::overflow()`, and `std::basic_streambuf<char_type, traits_type>::xsgetn()`.

**5.65.4.46** `template<typename _CharT, typename _Traits> streamsize  
std::basic_filebuf<_CharT, _Traits>::xspn( const char_type *  
__s, streamsize __n )` **[protected, virtual, inherited]**

Multiple character insertion.

### Parameters

- s* A buffer area.
- n* Maximum number of characters to write.

### Returns

The number of characters written.

Writes *s*[0] through *s*[*n*-1] to the output sequence, as if by `sputc()`. Stops when either *n* characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 641 of file `fstream.tcc`.

References `std::basic_filebuf<_CharT, _Traits>::_M_buf_size`, `std::basic_filebuf<_CharT, _Traits>::_M_mode`, `std::basic_filebuf<_CharT, _Traits>::_M_reading`, `std::basic_filebuf<_CharT, _Traits>::_M_set_buffer()`, `std::basic_streambuf<_CharT, _Traits>::eptr()`, `std::min()`, `std::ios_base::out`, `std::basic_streambuf<_CharT, _Traits>::pbase()`, `std::basic_streambuf<_CharT, _Traits>::pptr()`, and `std::basic_streambuf<char_type, traits_type>::xspn()`.

## 5.65.5 Member Data Documentation

**5.65.5.1** `template<typename _CharT, typename _Traits> char_type*  
std::basic_filebuf<_CharT, _Traits>::_M_buf` **[protected,  
inherited]**

Pointer to the beginning of internal buffer.

## 5.65 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 1027

Definition at line 111 of file `fstream`.

Referenced by `std::basic_filebuf<char_type, traits_type>::_M_destroy_pback()`, `std::basic_filebuf<char_type, traits_type>::_M_set_buffer()`, and `std::basic_filebuf<_CharT, _Traits>::setbuf()`.

### 5.65.5.2 `template<typename _CharT, typename _Traits> locale` `std::basic_streambuf<_CharT, _Traits>::_M_buf_locale` `[protected, inherited]`

Current locale setting.

Definition at line 190 of file `streambuf`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::basic_filebuf()`, `std::basic_streambuf<char_type, traits_type>::getloc()`, and `std::basic_streambuf<char_type, traits_type>::pubimbue()`.

### 5.65.5.3 `template<typename _CharT, typename _Traits> size_t` `std::basic_filebuf<_CharT, _Traits>::_M_buf_size` `[protected,` `inherited]`

Actual size of internal buffer. This number is equal to the size of the put area + 1 position, reserved for the overflow char of a full area.

Definition at line 118 of file `fstream`.

Referenced by `std::basic_filebuf<char_type, traits_type>::_M_set_buffer()`, `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::setbuf()`, `__gnu_cxx::stdio_filebuf<_CharT, _Traits>::stdio_filebuf()`, `std::basic_filebuf<_CharT, _Traits>::underflow()`, `std::basic_filebuf<_CharT, _Traits>::xsgetn()`, and `std::basic_filebuf<_CharT, _Traits>::xsputn()`.

### 5.65.5.4 `template<typename _CharT, typename _Traits> char*` `std::basic_filebuf<_CharT, _Traits>::_M_ext_buf` `[protected,` `inherited]`

Buffer for external characters. Used for input when `codecvt::always_noconv() == false`. When valid, this corresponds to [eback\(\)](#).

Definition at line 153 of file `fstream`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, and `std::basic_filebuf<_CharT, _Traits>::underflow()`.

**5.65.5.5** `template<typename _CharT, typename _Traits> streamsize  
std::basic_filebuf<_CharT, _Traits>::_M_ext_buf_size  
[protected, inherited]`

Size of buffer held by `_M_ext_buf`.

Definition at line 158 of file `fstream`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::underflow()`.

**5.65.5.6** `template<typename _CharT, typename _Traits> const char*  
std::basic_filebuf<_CharT, _Traits>::_M_ext_next [protected,  
inherited]`

Pointers into the buffer held by `_M_ext_buf` that delimit a subsequence of bytes that have been read but not yet converted. When valid, `_M_ext_next` corresponds to `egptr()`.

Definition at line 165 of file `fstream`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, and `std::basic_filebuf<_CharT, _Traits>::underflow()`.

**5.65.5.7** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::_M_in_beg  
[protected, inherited]`

This is based on `_IO_FILE`, just reordered to be more consistent, and is intended to be the most minimal abstraction for an internal buffer.

- `get == input == read`
- `put == output == write`

Definition at line 182 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::eback()`, and `std::basic_streambuf<char_type, traits_type>::setg()`.

**5.65.5.8** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::_M_in_cur  
[protected, inherited]`

Entry point for `imbue()`.

## 5.65 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 1029

---

### Parameters

*loc* The new locale.

### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 183 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::gbump()`, `std::basic_streambuf<char_type, traits_type>::gptr()`, and `std::basic_streambuf<char_type, traits_type>::setg()`.

**5.65.5.9** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::_M_in_end  
[protected, inherited]`

Entry point for [imbue\(\)](#).

### Parameters

*loc* The new locale.

### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 184 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::egptr()`, and `std::basic_streambuf<char_type, traits_type>::setg()`.

**5.65.5.10** `template<typename _CharT, typename _Traits> ios_base::openmode  
std::basic_filebuf<_CharT, _Traits>::_M_mode [protected,  
inherited]`

Place to stash in || out || in | out settings for current filebuf.

Definition at line 96 of file `fstream`.

Referenced by `std::basic_filebuf<char_type, traits_type>::_M_set_buffer()`, `std::basic_filebuf<_CharT, _Traits>::imbue()`, `std::basic_filebuf<_CharT, _Traits`



## 5.65 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 1030

`>::open()`, `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::pbackfail()`, `std::basic_filebuf<_CharT, _Traits>::showmanyc()`, `__gnu_cxx::stdio_filebuf<_CharT, _Traits>::stdio_filebuf()`, `std::basic_filebuf<_CharT, _Traits>::underflow()`, `std::basic_filebuf<_CharT, _Traits>::xsgetn()`, and `std::basic_filebuf<_CharT, _Traits>::xsputn()`.

**5.65.5.11** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::_M_out_beg  
[protected, inherited]`

Entry point for [imbue\(\)](#).

### Parameters

*loc* The new locale.

### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 185 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::pbase()`, and `std::basic_streambuf<char_type, traits_type>::setp()`.

**5.65.5.12** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::_M_out_cur  
[protected, inherited]`

Entry point for [imbue\(\)](#).

### Parameters

*loc* The new locale.

### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 186 of file `streambuf`.

## 5.65 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 1031

Referenced by `std::basic_streambuf< char_type, traits_type >::pbump()`, `std::basic_streambuf< char_type, traits_type >::pptr()`, and `std::basic_streambuf< char_type, traits_type >::setp()`.

**5.65.5.13** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_out_end  
[protected, inherited]`

Entry point for `imbue()`.

### Parameters

*loc* The new locale.

### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 187 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::epptr()`, and `std::basic_streambuf< char_type, traits_type >::setp()`.

**5.65.5.14** `template<typename _CharT, typename _Traits> char_type  
std::basic_filebuf< _CharT, _Traits >::_M_pback [protected,  
inherited]`

Necessary bits for putback buffer management.

### Note

pbacks of over one character are not currently supported.

Definition at line 139 of file `fstream`.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_create_pback()`.

**5.65.5.15** `template<typename _CharT, typename _Traits> char_type*  
std::basic_filebuf< _CharT, _Traits >::_M_pback_cur_save  
[protected, inherited]`

Necessary bits for putback buffer management.

## 5.65 \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits > Class Template Reference 1032

### Note

pbacks of over one character are not currently supported.

Definition at line 140 of file fstream.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_create_pback()`, and `std::basic_filebuf< char_type, traits_type >::_M_destroy_pback()`.

**5.65.5.16** `template<typename _CharT, typename _Traits> char_type*  
std::basic_filebuf< _CharT, _Traits >::_M_pback_end_save  
[protected, inherited]`

Necessary bits for putback buffer management.

### Note

pbacks of over one character are not currently supported.

Definition at line 141 of file fstream.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_create_pback()`, and `std::basic_filebuf< char_type, traits_type >::_M_destroy_pback()`.

**5.65.5.17** `template<typename _CharT, typename _Traits> bool  
std::basic_filebuf< _CharT, _Traits >::_M_pback_init  
[protected, inherited]`

Necessary bits for putback buffer management.

### Note

pbacks of over one character are not currently supported.

Definition at line 142 of file fstream.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_create_pback()`, `std::basic_filebuf< char_type, traits_type >::_M_destroy_pback()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

**5.65.5.18** `template<typename _CharT, typename _Traits> bool  
std::basic_filebuf< _CharT, _Traits >::_M_reading [protected,  
inherited]`

`_M_reading == false` && `_M_writing == false` for **uncommitted** mode; `_M_reading == true` for **read** mode; `_M_writing == true` for **write** mode;

NB: `_M_reading == true && _M_writing == true` is unused.

Definition at line 130 of file `fstream`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, `std::basic_filebuf<_CharT, _Traits>::open()`, `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::pbackfail()`, `std::basic_filebuf<_CharT, _Traits>::seekoff()`, `__gnu_cxx::stdio_filebuf<_CharT, _Traits>::stdio_filebuf()`, `std::basic_filebuf<_CharT, _Traits>::underflow()`, `std::basic_filebuf<_CharT, _Traits>::xsgetn()`, and `std::basic_filebuf<_CharT, _Traits>::xsputn()`.

The documentation for this class was generated from the following file:

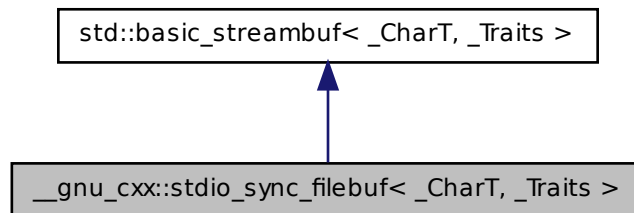
- [stdio\\_filebuf.h](#)

## **5.66 `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>` Class Template Reference**

Provides a layer of compatibility for C.

This GNU extension provides extensions for working with standard C `FILE*`'s. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.

Inheritance diagram for `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`:



### **Public Types**

- typedef `_CharT` [char\\_type](#)
- typedef `traits_type::int_type` [int\\_type](#)
- typedef `traits_type::off_type` [off\\_type](#)

- typedef traits\_type::pos\_type [pos\\_type](#)
- typedef \_Traits [traits\\_type](#)
- typedef basic\_streambuf< [char\\_type](#), [traits\\_type](#) > [\\_\\_streambuf\\_type](#)

### Public Member Functions

- **stdio\_sync\_filebuf** (std::\_\_c\_file \*\_\_f)
- std::\_\_c\_file \*const [file](#) ()
- streamsize [in\\_avail](#) ()
- [int\\_type](#) [sbumpc](#) ()
- [int\\_type](#) [sgetc](#) ()
- streamsize [sgetn](#) ([char\\_type](#) \*\_\_s, streamsize \_\_n)
- [int\\_type](#) [snextc](#) ()
- [int\\_type](#) [sputbackc](#) ([char\\_type](#) \_\_c)
- [int\\_type](#) [putc](#) ([char\\_type](#) \_\_c)
- streamsize [sputn](#) (const [char\\_type](#) \*\_\_s, streamsize \_\_n)
- [int\\_type](#) [sungetc](#) ()

### Protected Member Functions

- void [gbump](#) (int \_\_n)
- virtual void [imbue](#) (const locale &)
- virtual [int\\_type](#) [overflow](#) ([int\\_type](#) \_\_c=traits\_type::eof())
- virtual [int\\_type](#) [pbackfail](#) ([int\\_type](#) \_\_c=traits\_type::eof())
- void [pbump](#) (int \_\_n)
- virtual std::streampos [seekoff](#) (std::streamoff \_\_off, std::ios\_base::seekdir \_\_dir, std::ios\_base::openmode=std::ios\_base::in|std::ios\_base::out)
- virtual [pos\\_type](#) [seekoff](#) (off\_type, ios\_base::seekdir, ios\_base::openmode=ios\_base::in|ios\_base::out)
- virtual std::streampos [seekpos](#) (std::streampos \_\_pos, std::ios\_base::openmode \_\_mode=std::ios\_base::in|std::ios\_base::out)
- virtual [pos\\_type](#) [seekpos](#) ([pos\\_type](#), ios\_base::openmode=ios\_base::in|ios\_base::out)
- virtual basic\_streambuf< [char\\_type](#), \_Traits > \* [setbuf](#) ([char\\_type](#) \*, streamsize)
- void [setg](#) ([char\\_type](#) \*\_\_gbeg, [char\\_type](#) \*\_\_gnext, [char\\_type](#) \*\_\_gend)
- void [setp](#) ([char\\_type](#) \*\_\_pbeg, [char\\_type](#) \*\_\_pend)
- virtual streamsize [showmanyc](#) ()
- virtual int [sync](#) ()
- template<>  
[stdio\\_sync\\_filebuf](#)< wchar\_t >::[int\\_type](#) [syncgetc](#) ()
- template<>  
[stdio\\_sync\\_filebuf](#)< char >::[int\\_type](#) [syncgetc](#) ()

- `int_type syncgetc ()`
- `template<>`  
`stdio_sync_filebuf< wchar_t >::int_type syncputc (int_type __c)`
- `template<>`  
`stdio_sync_filebuf< char >::int_type syncputc (int_type __c)`
- `int_type syncputc (int_type __c)`
- `template<>`  
`stdio_sync_filebuf< wchar_t >::int_type syncungetc (int_type __c)`
- `int_type syncungetc (int_type __c)`
- `template<>`  
`stdio_sync_filebuf< char >::int_type syncungetc (int_type __c)`
- `virtual int_type uflow ()`
- `virtual int_type underflow ()`
- `template<>`  
`std::streamsize xsgetn (char *__s, std::streamsize __n)`
- `virtual std::streamsize xsgetn (char_type *__s, std::streamsize __n)`
- `template<>`  
`std::streamsize xsgetn (wchar_t *__s, std::streamsize __n)`
- `template<>`  
`std::streamsize xspn (const char *__s, std::streamsize __n)`
- `template<>`  
`std::streamsize xspn (const wchar_t *__s, std::streamsize __n)`
- `virtual std::streamsize xspn (const char_type *__s, std::streamsize __n)`
  
- `char_type * eback () const`
- `char_type * gptr () const`
- `char_type * egptr () const`
  
- `char_type * pbase () const`
- `char_type * pptr () const`
- `char_type * eptr () const`

## Friends

- `template<bool _IsMove, typename _CharT2 >`  
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, _CharT2 * >::__-`  
`type __copy_move_a2 (istreambuf_iterator< _CharT2 >, istreambuf_iterator<`  
`_CharT2 >, _CharT2 *)`
- `streamsize __copy_streambufs_eof (__streambuf_type *, __streambuf_type *,`  
`bool &)`
- `class basic_ios< char_type, traits_type >`
- `class basic_istream< char_type, traits_type >`
- `class basic_ostream< char_type, traits_type >`

- `template<typename _CharT2 >`  
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, istreambuf_`  
`iterator< _CharT2 > >::__type find (istreambuf_iterator< _CharT2 >,`  
`istreambuf_iterator< _CharT2 >, const _CharT2 &)`
- `template<typename _CharT2, typename _Traits2, typename _Alloc >`  
`basic_istream< _CharT2, _Traits2 > & getline (basic_istream< _CharT2, _`  
`Traits2 > &, basic_string< _CharT2, _Traits2, _Alloc > &, _CharT2)`
- `class istreambuf_iterator< char_type, traits_type >`
- `template<typename _CharT2, typename _Traits2, typename _Alloc >`  
`basic_istream< _CharT2, _Traits2 > & operator>> (basic_istream< _CharT2,`  
`_Traits2 > &, basic_string< _CharT2, _Traits2, _Alloc > &)`
- `template<typename _CharT2, typename _Traits2 >`  
`basic_istream< _CharT2, _Traits2 > & operator>> (basic_istream< _CharT2,`  
`_Traits2 > &, _CharT2 *)`
- `class ostreambuf_iterator< char_type, traits_type >`
  
- locale `pubimbue` (const locale &\_\_loc)
- locale `getloc` () const
- `__streambuf_type * pubsetbuf (char_type *__s, streamsize __n)`
- `pos_type pubseekoff (off_type __off, ios_base::seekdir __way, ios_`  
`base::openmode __mode=ios_base::in|ios_base::out)`
- `pos_type pubseekpos (pos_type __sp, ios_base::openmode __mode=ios_`  
`base::in|ios_base::out)`
- `int pubsync ()`
- `char_type * _M_in_beg`
- `char_type * _M_in_cur`
- `char_type * _M_in_end`
- `char_type * _M_out_beg`
- `char_type * _M_out_cur`
- `char_type * _M_out_end`
- locale `_M_buf_locale`

### 5.66.1 Detailed Description

`template<typename _CharT, typename _Traits = std::char_traits<_CharT>>`  
`class __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`

Provides a layer of compatibility for C.

This GNU extension provides extensions for working with standard C FILE\*'s. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.

Definition at line 57 of file `stdio_sync_filebuf.h`.

## 5.66.2 Member Typedef Documentation

**5.66.2.1** `template<typename _CharT, typename _Traits> typedef  
basic_streambuf<char_type, traits_type> std::basic_streambuf<  
_CharT, _Traits>::__streambuf_type [inherited]`

This is a non-standard type.

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, and `std::basic_filebuf<char_type, traits_type>`.

Definition at line 135 of file `streambuf`.

**5.66.2.2** `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> typedef _CharT __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::__char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 61 of file `stdio_sync_filebuf.h`.

**5.66.2.3** `template<typename _CharT, typename _Traits =  
std::char_traits<_CharT>> typedef traits_type::int_type  
__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 63 of file `stdio_sync_filebuf.h`.

**5.66.2.4** `template<typename _CharT, typename _Traits =  
std::char_traits<_CharT>> typedef traits_type::off_type  
__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.



Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits>](#).

Definition at line 65 of file `stdio_sync_filebuf.h`.

**5.66.2.5** `template<typename _CharT , typename _Traits =  
std::char_traits<_CharT>> typedef traits_type::pos_type  
__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits>](#).

Definition at line 64 of file `stdio_sync_filebuf.h`.

**5.66.2.6** `template<typename _CharT , typename _Traits = std::char_traits<_  
_CharT>> typedef _Traits __gnu_cxx::stdio_sync_filebuf<_CharT,  
_Traits>::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits>](#).

Definition at line 62 of file `stdio_sync_filebuf.h`.

### 5.66.3 Member Function Documentation

**5.66.3.1** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::eback ( ) const  
[inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- [eback\(\)](#) returns the beginning pointer for the input sequence
- [gptr\(\)](#) returns the next pointer for the input sequence
- [egptr\(\)](#) returns the end pointer for the input sequence

Definition at line 462 of file streambuf.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_destroy_pback()`, `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_streambuf< char_type, traits_type >::sputbackc()`, `std::basic_streambuf< char_type, traits_type >::sungetc()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

**5.66.3.2 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::egptr ( ) const`**  
**`[inline, protected, inherited]`**

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 468 of file streambuf.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_create_pback()`, `std::basic_streambuf< char_type, traits_type >::in_avail()`, `std::basic_streambuf< char_type, traits_type >::sbumpc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_streambuf< char_type, traits_type >::sgetc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::str()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, `std::basic_streambuf< _CharT, _Traits >::xsgetn()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

**5.66.3.3 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::eptr ( ) const`**  
**`[inline, protected, inherited]`**

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 515 of file `streambuf`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_streambuf<char_type, traits_type>::sputc()`, `std::basic_streambuf<_CharT, _Traits>::xspn()`, and `std::basic_filebuf<_CharT, _Traits>::xspn()`.

```
5.66.3.4 template<typename _CharT, typename _Traits =
std::char_traits<_CharT>> std::__c_file* const
__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::file ()
[inline]
```

### Returns

The underlying `FILE*`.

This function can be used to access the underlying C file pointer. Note that there is no way for the library to track what you do with the file, so be careful.

Definition at line 89 of file `stdio_sync_filebuf.h`.

```
5.66.3.5 template<typename _CharT, typename _Traits> void
std::basic_streambuf<_CharT, _Traits>::gbump (int __n)
[inline, protected, inherited]
```

Moving the read position.

### Parameters

*n* The delta by which to move.

This just advances the read position without returning any data.

Definition at line 478 of file `streambuf`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::pbackfail()`, `std::basic_filebuf<_CharT, _Traits>::pbackfail()`, `std::basic_streambuf<char_type, traits_type>::sbumpc()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, `std::basic_streambuf<char_type, traits_type>::sputbackc()`, `std::basic_streambuf<char_type, traits_type>::sungetc()`, `std::basic_streambuf<char_type, traits_type>::uflow()`, `std::basic_streambuf<_CharT, _Traits>::xsgetn()`, and `std::basic_filebuf<_CharT, _Traits>::xsgetn()`.

**5.66.3.6 `template<typename _CharT, typename _Traits> locale std::basic_streambuf<_CharT, _Traits>::getloc ( ) const [inline, inherited]`**

Locale access.

**Returns**

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 224 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::pubimbue()`.

**5.66.3.7 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::gptr ( ) const [inline, protected, inherited]`**

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 465 of file `streambuf`.

Referenced by `std::basic_filebuf<char_type, traits_type>::_M_create_pback()`, `std::basic_filebuf<char_type, traits_type>::_M_destroy_pback()`, `std::basic_filebuf<`

`_CharT, _Traits >::imbue()`, `std::basic_streambuf< char_type, traits_type >::in_avail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_streambuf< char_type, traits_type >::sbumpc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_streambuf< char_type, traits_type >::sgetc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_streambuf< char_type, traits_type >::sputbackc()`, `std::basic_streambuf< char_type, traits_type >::sungetc()`, `std::basic_streambuf< char_type, traits_type >::uflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, `std::basic_streambuf< _CharT, _Traits >::xsgetn()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

**5.66.3.8** `template<typename _CharT, typename _Traits> virtual void  
std::basic_streambuf< _CharT, _Traits >::imbue ( const locale & )  
[inline, protected, virtual, inherited]`

Changes translations.

#### Parameters

*loc* A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

#### Note

Base class version does nothing.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, and `std::basic_filebuf< char_type, traits_type >`.

Definition at line 556 of file streambuf.

Referenced by `std::basic_streambuf< char_type, traits_type >::pubimbue()`.

**5.66.3.9** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf< _CharT, _Traits >::in_avail ( ) [inline,  
inherited]`

Looking ahead into the stream.

### Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

Definition at line 264 of file `streambuf`.

```
5.66.3.10 template<typename _CharT , typename _Traits
 = std::char_traits<_CharT>> virtual int_type
 __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::overflow (
 int_type = traits_type::eof()) [inline, protected,
 virtual]
```

Consumes data from the buffer; writes to the controlled sequence.

### Parameters

*c* An additional character to consume.

### Returns

`eof()` to indicate failure, something else (usually *c*, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if *c* is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output `streambuf` can be created by overriding only this function (no buffer area will be used).

### Note

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 142 of file `stdio_sync_filebuf.h`.

**5.66.3.11** `template<typename _CharT, typename _Traits  
= std::char_traits<_CharT>> virtual int_type  
__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::pbackfail(  
int_type = traits_type::eof() ) [inline, protected,  
virtual]`

Tries to back up the input sequence.

#### Parameters

*c* The character to be inserted back into the sequence.

#### Returns

`eof()` on failure, *some other value* on success

#### Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

#### Note

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 117 of file `stdio_sync_filebuf.h`.

**5.66.3.12** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::pbase( ) const  
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 509 of file streambuf.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_filebuf<_CharT, _Traits>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::str()`, `std::basic_filebuf<_CharT, _Traits>::sync()`, and `std::basic_filebuf<_CharT, _Traits>::xsputn()`.

**5.66.3.13** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf<_CharT, _Traits>::pbump ( int __n )  
[inline, protected, inherited]`

Moving the write position.

#### Parameters

*n* The delta by which to move.

This just advances the write position without returning any data.

Definition at line 525 of file streambuf.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, `std::basic_streambuf<char_type, traits_type>::sputc()`, and `std::basic_streambuf<_CharT, _Traits>::xsputn()`.

**5.66.3.14** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::pptr ( ) const  
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence



Definition at line 512 of file `streambuf`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_filebuf<_CharT, _Traits>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, `std::basic_streambuf<char_type, traits_type>::sputc()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::str()`, `std::basic_filebuf<_CharT, _Traits>::sync()`, `std::basic_streambuf<_CharT, _Traits>::xsputn()`, and `std::basic_filebuf<_CharT, _Traits>::xsputn()`.

**5.66.3.15** `template<typename _CharT, typename _Traits> locale  
std::basic_streambuf<_CharT, _Traits>::pubimbue ( const locale  
& __loc ) [inline, inherited]`

Entry point for `imbue()`.

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 207 of file `streambuf`.

**5.66.3.16** `template<typename _CharT, typename _Traits> pos_type  
std::basic_streambuf<_CharT, _Traits>::pubseekoff ( off_type  
__off, ios_base::seekdir __way, ios_base::openmode __mode =  
ios_base::in | ios_base::out ) [inline, inherited]`

Entry point for `imbue()`.

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 241 of file `streambuf`.

**5.66.3.17** `template<typename _CharT, typename _Traits> pos_type  
std::basic_streambuf<_CharT, _Traits>::pubseekpos ( pos_type  
__sp, ios_base::openmode __mode = ios_base::in | ios_base::out )  
[inline, inherited]`

Entry point for `imbue()`.

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 246 of file `streambuf`.

**5.66.3.18** `template<typename _CharT, typename _Traits> __streambuf_type*  
std::basic_streambuf<_CharT, _Traits>::pubsetbuf ( char_type *  
__s, streamsize __n ) [inline, inherited]`

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 237 of file `streambuf`.

**5.66.3.19** `template<typename _CharT, typename _Traits> int  
std::basic_streambuf<_CharT, _Traits>::pubsync ( ) [inline,  
inherited]`

Entry point for `imbue()`.

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 251 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::sync()`.

**5.66.3.20** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf<_CharT, _Traits>::sbumpc ( ) [inline,  
inherited]`

Getting the next character.

#### Returns

The next character, or `eof`.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 296 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::istreambuf_iterator<_CharT, _Traits>::operator++()`, and `std::basic_streambuf<char_type, traits_type>::snextc()`.

**5.66.3.21** `template<typename _CharT, typename _Traits> virtual  
pos_type std::basic_streambuf<_CharT, _Traits>::seekoff  
( off_type, ios_base::seekdir, ios_base::openmode =  
ios_base::in | ios_base::out ) [inline, protected,  
virtual, inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

#### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, and `std::basic_filebuf<char_type, traits_type>`.

Definition at line 582 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::pubseekoff()`.

**5.66.3.22** `template<typename _CharT, typename _Traits> virtual pos_type  
std::basic_streambuf<_CharT, _Traits>::seekpos ( pos_type,  
ios_base::openmode = ios_base::in | ios_base::out ) [inline,  
protected, virtual, inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, and `std::basic_filebuf<char_type, traits_type>`.

Definition at line 594 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::pubseekpos()`.

**5.66.3.23** `template<typename _CharT, typename _Traits> virtual  
basic_streambuf<char_type, _Traits>* std::basic_streambuf<  
_CharT, _Traits>::setbuf ( char_type*, streamsize ) [inline,  
protected, virtual, inherited]`

Manipulates the buffer.

Each derived class provides its own appropriate behavior. See the next-to-last paragraph of <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more on this function.

**Note**

Base class version does nothing, returns `this`.

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, and `std::basic_filebuf<char_type, traits_type>`.

Definition at line 571 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::pubsetbuf()`.

**5.66.3.24** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf<_CharT, _Traits>::setg ( char_type *  
__gbeg, char_type * __gnext, char_type * __gend ) [inline,  
protected, inherited]`

Setting the three read area pointers.

**Parameters**

*gbeg* A pointer.

*gnext* A pointer.

*gend* A pointer.

**Postcondition**

*gbeg* == `eback()`, *gnext* == `gptr()`, and *gend* == `egptr()`

Definition at line 489 of file `streambuf`.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_create_pback()`, `std::basic_filebuf< char_type, traits_type >::_M_destroy_pback()`, and `std::basic_filebuf< char_type, traits_type >::_M_set_buffer()`.

**5.66.3.25** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf<_CharT, _Traits>::setp ( char_type *  
__pbeg, char_type * __pend ) [inline, protected,  
inherited]`

Setting the three write area pointers.

**Parameters**

*pbeg* A pointer.

*pend* A pointer.

**Postcondition**

*pbeg* == `pbase()`, *pbeg* == `pptr()`, and *pend* == `pptr()`

Definition at line 535 of file `streambuf`.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_set_buffer()`.

**5.66.3.26** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf<_CharT, _Traits>::sgetc ( ) [inline,  
inherited]`

Getting the next character.

#### Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 318 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, and `std::basic_streambuf<char_type, traits_type>::snextc()`.

**5.66.3.27** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf<_CharT, _Traits>::sgetn ( char_type * __s,  
streamsize __n ) [inline, inherited]`

Entry point for `xsgetn`.

#### Parameters

*s* A buffer area.

*n* A count.

Returns `xsgetn(s,n)`. The effect is to fill `s[0]` through `s[n-1]` with characters from the input sequence, if possible.

Definition at line 337 of file `streambuf`.

**5.66.3.28** `template<typename _CharT, typename _Traits> virtual streamsize  
std::basic_streambuf<_CharT, _Traits>::showmanyc ( )  
[inline, protected, virtual, inherited]`

Investigating the data available.

### Returns

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.*  
[27.5.2.4.3]/1

### Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, and `std::basic_filebuf<char_type, traits_type>`.

Definition at line 629 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::in_avail()`.

**5.66.3.29** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf<_CharT, _Traits>::snextc ( ) [inline,  
inherited]`

Getting the next character.

### Returns

The next character, or `eof`.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 278 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

**5.66.3.30** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf<_CharT, _Traits>::sputbackc ( char_type  
__c ) [inline, inherited]`

Pushing characters back into the input stream.

#### Parameters

*c* The character to push back.

#### Returns

The previous character, if possible.

Similar to [sungetc\(\)](#), but *c* is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be *c*.

Definition at line 352 of file streambuf.

Referenced by `std::basic_istream<_CharT, _Traits>::putback()`.

**5.66.3.31** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf<_CharT, _Traits>::sputc ( char_type __c )  
[inline, inherited]`

Entry point for all single-character output functions.

#### Parameters

*c* A character to output.

#### Returns

*c*, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores *c* in that position, increments the position, and returns `traits::to_int_type(c)`. If a write position is not available, returns `overflow(c)`.

Definition at line 404 of file streambuf.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, and `std::ostreambuf_iterator<_CharT, _Traits>::operator=()`.



**5.66.3.32** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf<_CharT, _Traits>::sputn ( const char_type *  
__s, streamsize __n ) [inline, inherited]`

Entry point for all single-character output functions.

#### Parameters

*s* A buffer read area.

*n* A count.

One of two public output functions.

Returns `xsputn(s,n)`. The effect is to write `s[0]` through `s[n-1]` to the output sequence, if possible.

Definition at line 430 of file `streambuf`.

**5.66.3.33** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf<_CharT, _Traits>::sungetc ( ) [inline,  
inherited]`

Moving backwards in the input stream.

#### Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbackfail()`. The effect is to *unget* the last character *gotten*.

Definition at line 377 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::unget()`.

**5.66.3.34** `template<typename _CharT, typename _Traits = std::char_traits<_  
CharT>> virtual int __gnu_cxx::stdio_sync_filebuf<_CharT,  
_Traits>::sync ( void ) [inline, protected, virtual]`

Synchronizes the buffer arrays with the controlled sequences.

### Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

### Note

Base class version does nothing, returns zero.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 161 of file `stdio_sync_filebuf.h`.

```
5.66.3.35 template<typename _CharT , typename _Traits
 = std::char_traits<_CharT>> virtual int_type
 __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::uflow ()
 [inline, protected, virtual]
```

Fetches more data from the controlled sequence.

### Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 109 of file `stdio_sync_filebuf.h`.

```
5.66.3.36 template<typename _CharT , typename _Traits
 = std::char_traits<_CharT>> virtual int_type
 __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::underflow ()
 [inline, protected, virtual]
```

Fetches more data from the controlled sequence.

### Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25.html>

#### Note

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 102 of file `stdio_sync_filebuf.h`.

```
5.66.3.37 template<typename _CharT , typename _Traits =
 std::char_traits<_CharT>> virtual std::streamsize
 __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::xsgetn (
 char_type * __s, std::streamsize __n) [protected, virtual]
```

Multiple character extraction.

#### Parameters

*s* A buffer area.

*n* Maximum number of characters to assign.

#### Returns

The number of characters assigned.

Fills `s[0]` through `s[n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either *n* characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

**5.66.3.38** `template<typename _CharT, typename _Traits =  
std::char_traits<_CharT>> virtual std::streamsize  
__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::xspn( const  
char_type* __s, std::streamsize __n ) [protected, virtual]`

Multiple character insertion.

#### Parameters

- s* A buffer area.
- n* Maximum number of characters to write.

#### Returns

The number of characters written.

Writes *s*[0] through *s*[*n*-1] to the output sequence, as if by `sputc()`. Stops when either *n* characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

#### 5.66.4 Member Data Documentation

**5.66.4.1** `template<typename _CharT, typename _Traits> locale  
std::basic_streambuf<_CharT, _Traits>::_M_buf_locale  
[protected, inherited]`

Current locale setting.

Definition at line 190 of file `streambuf`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::basic_filebuf()`, `std::basic_streambuf<char_type, traits_type>::getloc()`, and `std::basic_streambuf<char_type, traits_type>::pubimbue()`.

**5.66.4.2** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::_M_in_beg  
[protected, inherited]`

This is based on `_IO_FILE`, just reordered to be more consistent, and is intended to be the most minimal abstraction for an internal buffer.

- `get == input == read`
- `put == output == write`

Definition at line 182 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::eback()`, and `std::basic_streambuf< char_type, traits_type >::setg()`.

**5.66.4.3** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_in_cur  
[protected, inherited]`

Entry point for [imbue\(\)](#).

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 183 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::gbump()`, `std::basic_streambuf< char_type, traits_type >::gptr()`, and `std::basic_streambuf< char_type, traits_type >::setg()`.

**5.66.4.4** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_in_end  
[protected, inherited]`

Entry point for [imbue\(\)](#).

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 184 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::egptr()`, and `std::basic_streambuf< char_type, traits_type >::setg()`.

**5.66.4.5** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_out_beg  
[protected, inherited]`

Entry point for `imbue()`.

**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 185 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::pbase()`, and `std::basic_streambuf< char_type, traits_type >::setp()`.

**5.66.4.6** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_out_cur  
[protected, inherited]`

Entry point for `imbue()`.

**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 186 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::pbump()`, `std::basic_streambuf< char_type, traits_type >::pptr()`, and `std::basic_streambuf< char_type, traits_type >::setp()`.

**5.66.4.7** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_out_end  
[protected, inherited]`

Entry point for [imbue\(\)](#).

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 187 of file `streambuf`.

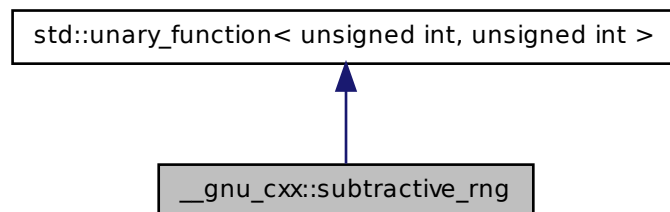
Referenced by `std::basic_streambuf< char_type, traits_type >::epptr()`, and `std::basic_streambuf< char_type, traits_type >::setp()`.

The documentation for this class was generated from the following file:

- [stdio\\_sync\\_filebuf.h](#)

## 5.67 `__gnu_cxx::subtractive_rng` Class Reference

Inheritance diagram for `__gnu_cxx::subtractive_rng`:



### Public Types

- typedef `_Arg` [argument\\_type](#)
- typedef `_Result` [result\\_type](#)

### Public Member Functions

- [subtractive\\_rng](#) (unsigned int `__seed`)
- [subtractive\\_rng](#) ()
- void `_M_initialize` (unsigned int `__seed`)
- unsigned int [operator\(\)](#) (unsigned int `__limit`)

#### 5.67.1 Detailed Description

The [subtractive\\_rng](#) class is documented on [SGI's site](#). Note that this code assumes that `int` is 32 bits.

Definition at line 349 of file `ext/functional`.



### 5.67.2 Member Typedef Documentation

**5.67.2.1** `template<typename _Arg, typename _Result> typedef _Arg  
std::unary_function< _Arg, _Result >::argument_type  
[inherited]`

`argument_type` is the type of the argument

Definition at line 105 of file `stl_function.h`.

**5.67.2.2** `template<typename _Arg, typename _Result> typedef _Result  
std::unary_function< _Arg, _Result >::result_type [inherited]`

`result_type` is the return type

Definition at line 108 of file `stl_function.h`.

### 5.67.3 Constructor & Destructor Documentation

**5.67.3.1** `__gnu_cxx::subtractive_rng::subtractive_rng ( unsigned int __seed )  
[inline]`

Ctor allowing you to initialize the seed.

Definition at line 391 of file `ext/functional`.

**5.67.3.2** `__gnu_cxx::subtractive_rng::subtractive_rng ( ) [inline]`

Default ctor; initializes its state with some number you don't see.

Definition at line 395 of file `ext/functional`.

### 5.67.4 Member Function Documentation

**5.67.4.1** `unsigned int __gnu_cxx::subtractive_rng::operator() ( unsigned int  
__limit ) [inline]`

Returns a number less than the argument.

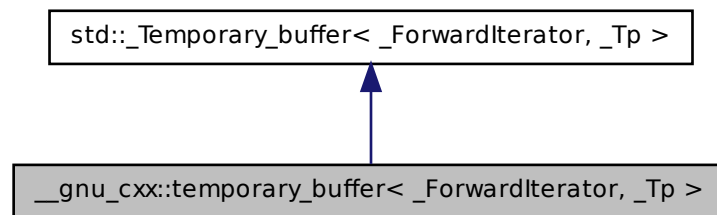
Definition at line 360 of file `ext/functional`.

The documentation for this class was generated from the following file:

- [ext/functional](#)

## 5.68 `__gnu_cxx::temporary_buffer<_ForwardIterator, _Tp>` Struct Template Reference

Inheritance diagram for `__gnu_cxx::temporary_buffer<_ForwardIterator, _Tp>`:



### Public Types

- typedef pointer **iterator**
- typedef value\_type \* **pointer**
- typedef ptrdiff\_t **size\_type**
- typedef \_Tp **value\_type**

### Public Member Functions

- [temporary\\_buffer](#) (`_ForwardIterator __first, _ForwardIterator __last`)
- [~temporary\\_buffer](#) ()
- [iterator begin](#) ()
- [iterator end](#) ()
- size\_type [requested\\_size](#) () const
- size\_type [size](#) () const

## Protected Attributes

- pointer `_M_buffer`
- size\_type `_M_len`
- size\_type `_M_original_len`

### 5.68.1 Detailed Description

```
template<class _ForwardIterator, class _Tp = typename std::iterator_traits<_ForwardIterator>::value_type> struct __gnu_cxx::temporary_buffer<_ForwardIterator, _Tp>
```

This class provides similar behavior and semantics of the standard functions [get\\_temporary\\_buffer\(\)](#) and [return\\_temporary\\_buffer\(\)](#), but encapsulated in a type vaguely resembling a standard container.

By default, a `temporary_buffer<Iter>` stores space for objects of whatever type the `Iter` iterator points to. It is constructed from a typical `[first,last)` range, and provides the [begin\(\)](#), [end\(\)](#), [size\(\)](#) functions, as well as [requested\\_size\(\)](#). For non-trivial types, copies of `*first` will be used to initialize the storage.

`malloc` is used to obtain underlying storage.

Like [get\\_temporary\\_buffer\(\)](#), not all the requested memory may be available. Ideally, the created buffer will be large enough to hold a copy of `[first,last)`, but if [size\(\)](#) is less than [requested\\_size\(\)](#), then this didn't happen.

Definition at line 184 of file `ext/memory`.

### 5.68.2 Constructor & Destructor Documentation

```
5.68.2.1 template<class _ForwardIterator , class _Tp = typename
std::iterator_traits<_ForwardIterator>::value_type>
__gnu_cxx::temporary_buffer<_ForwardIterator, _Tp
>::temporary_buffer (_ForwardIterator __first, _ForwardIterator
__last) [inline]
```

Requests storage large enough to hold a copy of `[first,last)`.

Definition at line 187 of file `ext/memory`.

**5.68.2.2** `template<class _ForwardIterator, class _Tp = typename  
std::iterator_traits<_ForwardIterator>::value_type>  
__gnu_cxx::temporary_buffer<_ForwardIterator, _Tp  
>::~~temporary_buffer ( ) [inline]`

Destroys objects and frees storage.

Definition at line 191 of file `ext/memory`.

### 5.68.3 Member Function Documentation

**5.68.3.1** `template<typename _ForwardIterator, typename _Tp> iterator  
std::_Temporary_buffer<_ForwardIterator, _Tp>::begin ( )  
[inline, inherited]`

As per Table mumble.

Definition at line 152 of file `stl_tempbuf.h`.

Referenced by `std::inplace_merge()`, `std::stable_partition()`, and `std::stable_sort()`.

**5.68.3.2** `template<typename _ForwardIterator, typename _Tp> iterator  
std::_Temporary_buffer<_ForwardIterator, _Tp>::end ( )  
[inline, inherited]`

As per Table mumble.

Definition at line 157 of file `stl_tempbuf.h`.

**5.68.3.3** `template<typename _ForwardIterator, typename _Tp> size_type  
std::_Temporary_buffer<_ForwardIterator, _Tp>::requested_size ( ) const [inline, inherited]`

Returns the size requested by the constructor; may be `>size()`.

Definition at line 147 of file `stl_tempbuf.h`.

Referenced by `std::stable_partition()`.

5.68.3.4 `template<typename _ForwardIterator, typename _Tp> size_type  
std::_Temporary_buffer< _ForwardIterator, _Tp >::size ( ) const  
[inline, inherited]`

As per Table mumble.

Definition at line 142 of file `stl_tempbuf.h`.

Referenced by `std::inplace_merge()`, `std::stable_partition()`, and `std::stable_sort()`.

The documentation for this struct was generated from the following file:

- [ext/memory](#)

## 5.69 `__gnu_cxx::throw_allocator_base< _Tp, _Cond >` Class Template Reference

Allocator class with logging and exception generation control. Intended to be used as an `allocator_type` in templated code.

Note: Deallocate not allowed to throw.

Inheritance diagram for `__gnu_cxx::throw_allocator_base< _Tp, _Cond >`:



### Public Types

- `typedef const value_type * const_pointer`
- `typedef const value_type & const_reference`
- `typedef ptrdiff_t difference_type`
- `typedef value_type * pointer`
- `typedef value_type & reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

### Public Member Functions

- `pointer address (reference __x) const`
- `const_pointer address (const_reference __x) const`

## 5.70 `__gnu_cxx::throw_allocator_limit< _Tp >` Struct Template Reference 067

- pointer **allocate** (size\_type \_\_n, [std::allocator](#)< void >::const\_pointer hint=0)
- void **check\_allocated** (pointer \_\_p, size\_type \_\_n)
- void **check\_allocated** (size\_type \_\_n)
- void **check\_allocated** (void \*p, size\_t size)
- void **check\_allocated** (size\_t label)
- void **construct** (pointer \_\_p, const value\_type &val)
- template<typename... \_Args>  
void **construct** (pointer \_\_p, \_Args &&...\_\_args)
- void **deallocate** (pointer \_\_p, size\_type \_\_n)
- void **destroy** (pointer \_\_p)
- void **erase** (void \*p, size\_t size)
- void **insert** (void \*p, size\_t size)
- size\_type **max\_size** () const throw ()

### Static Public Member Functions

- static size\_t **get\_label** ()
- static void **set\_label** (size\_t l)

#### 5.69.1 Detailed Description

`template<typename _Tp, typename _Cond> class __gnu_cxx::throw_allocator_base< _Tp, _Cond >`

Allocator class with logging and exception generation control. Intended to be used as an allocator\_type in templated code.

Note: Deallocate not allowed to throw.

Definition at line 600 of file `throw_allocator.h`.

The documentation for this class was generated from the following file:

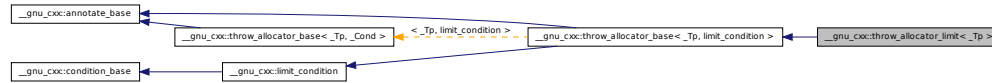
- [throw\\_allocator.h](#)

## 5.70 `__gnu_cxx::throw_allocator_limit< _Tp >` Struct Template Reference

Allocator throwing via limit condition.

## 5.70 `__gnu_cxx::throw_allocator_limit<_Tp>` Struct Template Reference 068

Inheritance diagram for `__gnu_cxx::throw_allocator_limit<_Tp>`:



### Public Types

- `typedef const value_type * const_pointer`
- `typedef const value_type & const_reference`
- `typedef ptrdiff_t difference_type`
- `typedef value_type * pointer`
- `typedef value_type & reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

### Public Member Functions

- `throw_allocator_limit` (const `throw_allocator_limit` &) `throw ()`
- `template<typename _Tp1 > throw_allocator_limit` (const `throw_allocator_limit`<\_Tp1 > &) `throw ()`
- `pointer address` (reference \_\_x) `const`
- `const_pointer address` (const\_reference \_\_x) `const`
- `pointer allocate` (size\_type \_\_n, `std::allocator`< void >::const\_pointer hint=0)
- `void check_allocated` (size\_t label)
- `void check_allocated` (size\_type \_\_n)
- `void check_allocated` (pointer \_\_p, size\_type \_\_n)
- `void check_allocated` (void \*p, size\_t size)
- `void construct` (pointer \_\_p, const value\_type &val)
- `void construct` (pointer \_\_p, \_Args &&...\_\_args)
- `void deallocate` (pointer \_\_p, size\_type \_\_n)
- `void destroy` (pointer \_\_p)
- `void erase` (void \*p, size\_t size)
- `void insert` (void \*p, size\_t size)
- `size_type max_size` () `const throw ()`

## 5.71 `__gnu_cxx::throw_allocator_random<_Tp>` Struct Template Reference

### Static Public Member Functions

- static `size_t` & **count** ()
- static `size_t` **get\_label** ()
- static `size_t` & **limit** ()
- static void **set\_label** (`size_t` l)
- static void **set\_limit** (`const size_t` \_\_l)
- static void **throw\_conditionally** ()

#### 5.70.1 Detailed Description

`template<typename _Tp> struct __gnu_cxx::throw_allocator_limit<_Tp>`

Allocator throwing via limit condition.

Definition at line 690 of file `throw_allocator.h`.

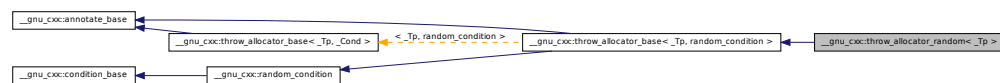
The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.71 `__gnu_cxx::throw_allocator_random<_Tp>` Struct Template Reference

Allocator throwing via random condition.

Inheritance diagram for `__gnu_cxx::throw_allocator_random<_Tp>`:



### Public Types

- `typedef const value_type * const_pointer`
- `typedef const value_type & const_reference`
- `typedef ptrdiff_t difference_type`
- `typedef value_type * pointer`
- `typedef value_type & reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`



## Public Member Functions

- **throw\_allocator\_random** (const [throw\\_allocator\\_random](#) &) throw ()
- `template<typename _Tp1 >`  
**throw\_allocator\_random** (const [throw\\_allocator\\_random](#)<\_Tp1> &) throw ()
- pointer **address** (reference \_\_x) const
- const\_pointer **address** (const\_reference \_\_x) const
- pointer **allocate** (size\_type \_\_n, [std::allocator](#)< void >::const\_pointer hint=0)
- void **check\_allocated** (size\_t label)
- void **check\_allocated** (void \*p, size\_t size)
- void **check\_allocated** (pointer \_\_p, size\_type \_\_n)
- void **check\_allocated** (size\_type \_\_n)
- void **construct** (pointer \_\_p, const value\_type &val)
- void **construct** (pointer \_\_p, \_Args &&...\_\_args)
- void **deallocate** (pointer \_\_p, size\_type \_\_n)
- void **destroy** (pointer \_\_p)
- void **erase** (void \*p, size\_t size)
- void **insert** (void \*p, size\_t size)
- size\_type **max\_size** () const throw ()
- void **seed** (unsigned long \_\_s)

## Static Public Member Functions

- static size\_t **get\_label** ()
- static void **set\_label** (size\_t l)
- static void **set\_probability** (double \_\_p)
- static void **throw\_conditionally** ()

### 5.71.1 Detailed Description

`template<typename _Tp> struct __gnu_cxx::throw_allocator_random<_Tp>`

Allocator throwing via random condition.

Definition at line 709 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

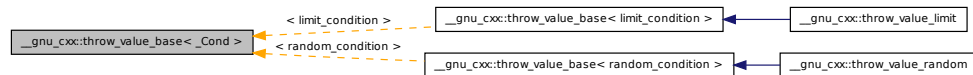
- [throw\\_allocator.h](#)

## 5.72 `__gnu_cxx::throw_value_base<_Cond>` Struct Template Reference 1071

### 5.72 `__gnu_cxx::throw_value_base<_Cond>` Struct Template Reference

Class with exception generation control. Intended to be used as a `value_type` in templated code.

Inheritance diagram for `__gnu_cxx::throw_value_base<_Cond>`:



#### Public Types

- `typedef _Cond condition_type`

#### Public Member Functions

- `throw_value_base` (const [throw\\_value\\_base](#) &\_\_v)
- `throw_value_base` (const `std::size_t` \_\_i)
- [throw\\_value\\_base](#) & `operator++` ()
- [throw\\_value\\_base](#) & `operator=` (const [throw\\_value\\_base](#) &\_\_v)

#### Public Attributes

- `std::size_t _M_i`

#### 5.72.1 Detailed Description

`template<typename _Cond> struct __gnu_cxx::throw_value_base<_Cond>`

Class with exception generation control. Intended to be used as a `value_type` in templated code. Note: Destructor not allowed to throw.

Definition at line 455 of file `throw_allocator.h`.

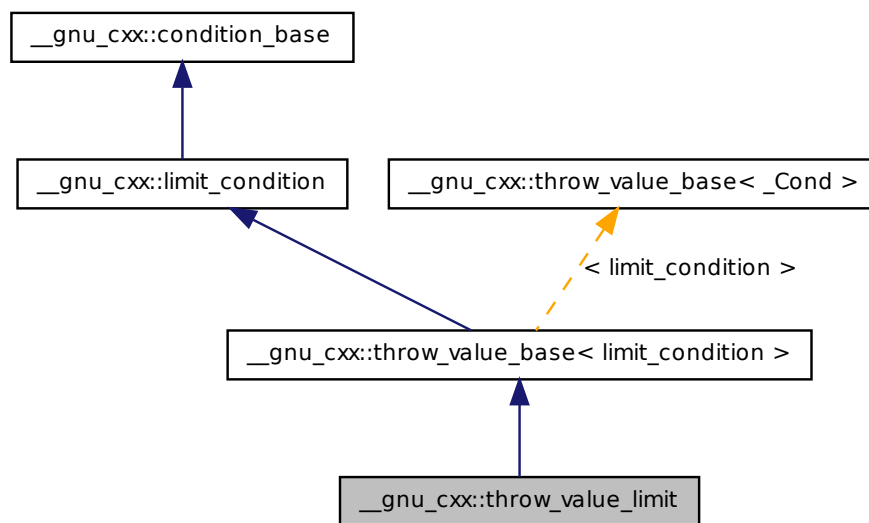
The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

### 5.73 \_\_gnu\_cxx::throw\_value\_limit Struct Reference

Type throwing via limit condition.

Inheritance diagram for \_\_gnu\_cxx::throw\_value\_limit:



#### Public Types

- typedef [throw\\_value\\_base< limit\\_condition >](#) **base\_type**
- typedef [limit\\_condition](#) **condition\_type**

#### Public Member Functions

- **throw\_value\_limit** (const [throw\\_value\\_limit](#) &\_\_other)
- **throw\_value\_limit** (const std::size\_t \_\_i)
- [throw\\_value\\_base](#) & **operator++** ()

#### Static Public Member Functions

- static size\_t & **count** ()

- static `size_t` & **limit** ()
- static void **set\_limit** (const `size_t` \_\_l)
- static void **throw\_conditionally** ()

#### Public Attributes

- `std::size_t` **M\_i**

##### 5.73.1 Detailed Description

Type throwing via limit condition.

Definition at line 561 of file `throw_allocator.h`.

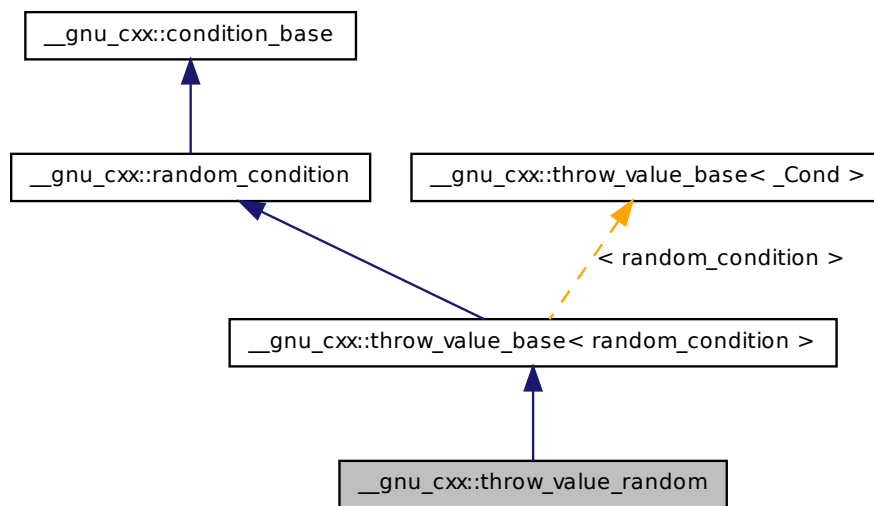
The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

#### 5.74 `__gnu_cxx::throw_value_random` Struct Reference

Type throwing via random condition.

Inheritance diagram for \_\_gnu\_cxx::throw\_value\_random:



### Public Types

- typedef `throw_value_base< random_condition >` `base_type`
- typedef `random_condition` `condition_type`

### Public Member Functions

- `throw_value_random` (const `throw_value_random` &\_\_other)
- `throw_value_random` (const std::size\_t \_\_i)
- `throw_value_base` & `operator++` ()
- void `seed` (unsigned long \_\_s)

### Static Public Member Functions

- static void `set_probability` (double \_\_p)
- static void `throw_conditionally` ()

## Public Attributes

- `std::size_t _M_i`

### 5.74.1 Detailed Description

Type throwing via random condition.

Definition at line 576 of file `throw_allocator.h`.

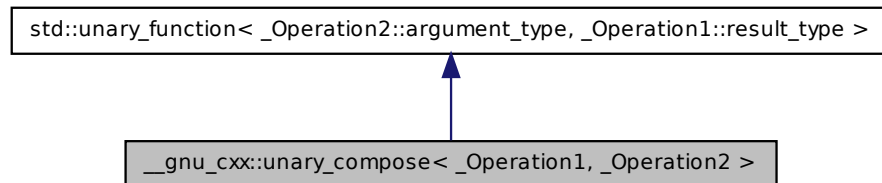
The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.75 `__gnu_cxx::unary_compose< _Operation1, _Operation2 >` Class Template Reference

An [SGI extension](#).

Inheritance diagram for `__gnu_cxx::unary_compose< _Operation1, _Operation2 >`:



## Public Types

- typedef `_Arg` [argument\\_type](#)
- typedef `_Result` [result\\_type](#)

## Public Member Functions

- **unary\_compose** (`const _Operation1 &__x, const _Operation2 &__y`)
- `_Operation1::result_type` **operator()** (`const typename _Operation2::argument_type &__x`) const

## 5.76 `__gnu_debug::After_nth_from<_Iterator>` Class Template Reference

### Protected Attributes

- `_Operation1 _M_fn1`
- `_Operation2 _M_fn2`

### 5.75.1 Detailed Description

`template<class _Operation1, class _Operation2> class __gnu_cxx::unary_compose<_Operation1, _Operation2>`

An [SGI extension](#).

Definition at line 126 of file `ext/functional`.

### 5.75.2 Member Typedef Documentation

**5.75.2.1** `template<typename _Arg, typename _Result> typedef _Arg std::unary_function<_Arg, _Result>::argument_type [inherited]`

`argument_type` is the type of the argument

Definition at line 105 of file `stl_function.h`.

**5.75.2.2** `template<typename _Arg, typename _Result> typedef _Result std::unary_function<_Arg, _Result>::result_type [inherited]`

`result_type` is the return type

Definition at line 108 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [ext/functional](#)

## 5.76 `__gnu_debug::After_nth_from<_Iterator>` Class Template Reference

### Public Member Functions

- `_After_nth_from` (const `difference_type` &\_\_n, const `_Iterator` &\_\_base)
- `bool operator()` (const `_Iterator` &\_\_x) const

### 5.76.1 Detailed Description

`template<typename _Iterator> class __gnu_debug::_After_nth_from<_Iterator>`

A function object that returns true when the given random access iterator is at least `n` steps away from the given iterator.

Definition at line 78 of file `safe_sequence.h`.

The documentation for this class was generated from the following file:

- [safe\\_sequence.h](#)

## 5.77 `__gnu_debug::_BeforeBeginHelper<_Sequence>` Struct Template Reference

### Public Types

- `typedef _It::iterator_type _BaseIt`
- `typedef _Sequence::const_iterator _It`

### Static Public Member Functions

- `static bool _M_Is(_BaseIt __it, const _Sequence *__seq)`

### 5.77.1 Detailed Description

`template<typename _Sequence> struct __gnu_debug::_BeforeBeginHelper<_Sequence>`

Helper struct to deal with sequence offering a `before_begin` iterator.

Definition at line 47 of file `safe_iterator.h`.

The documentation for this struct was generated from the following file:

- [safe\\_iterator.h](#)

## 5.78 `__gnu_debug::_Equal_to<_Type>` Class Template Reference

### Public Member Functions

- `_Equal_to(const _Type &__v)`



## 5.79 `__gnu_debug::_Not_equal_to<_Type>` Class Template Reference 1078

---

- `bool operator() (const _Type &__x) const`

### 5.78.1 Detailed Description

`template<typename _Type> class __gnu_debug::_Equal_to<_Type>`

A simple function object that returns true if the passed-in value is equal to the stored value.

Definition at line 63 of file `safe_sequence.h`.

The documentation for this class was generated from the following file:

- [safe\\_sequence.h](#)

## 5.79 `__gnu_debug::_Not_equal_to<_Type>` Class Template Reference

### Public Member Functions

- `_Not_equal_to (const _Type &__v)`
- `bool operator() (const _Type &__x) const`

### 5.79.1 Detailed Description

`template<typename _Type> class __gnu_debug::_Not_equal_to<_Type>`

A simple function object that returns true if the passed-in value is not equal to the stored value. It saves typing over using both `bind1st` and `not_equal`.

Definition at line 48 of file `safe_sequence.h`.

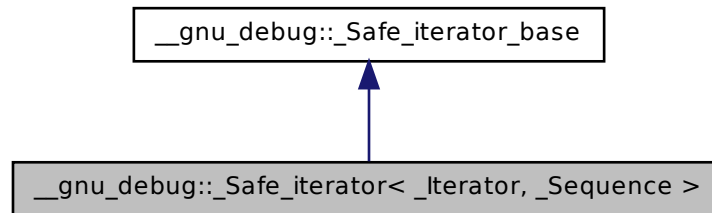
The documentation for this class was generated from the following file:

- [safe\\_sequence.h](#)

## 5.80 `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>` Class Template Reference

Safe iterator wrapper.

Inheritance diagram for `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>`:



### Public Types

- typedef `_Traits::difference_type` **difference\_type**
- typedef `_Traits::iterator_category` **iterator\_category**
- typedef `_Iterator` **iterator\_type**
- typedef `_Traits::pointer` **pointer**
- typedef `_Traits::reference` **reference**
- typedef `_Traits::value_type` **value\_type**

### Public Member Functions

- [`\_Safe\_iterator`](#) ()
- [`\_Safe\_iterator`](#) (const `_Iterator` &`__i`, const `_Sequence` \*`__seq`)
- template<typename `_MutableIterator` >  
[`\_Safe\_iterator`](#) (const [`\_Safe\_iterator`](#)< `_MutableIterator`, typename `__gnu_cxx::__enable_if`<(std::\_\_are\_same< `_MutableIterator`, typename `_Sequence::iterator::iterator_type` >::\_\_value), `_Sequence` >::\_\_type > &`__x`)
- [`\_Safe\_iterator`](#) (const [`\_Safe\_iterator`](#) &`__x`)
- void [`\_M\_attach`](#) ([`\_Safe\_sequence\_base`](#) \*`__seq`)
- void [`\_M\_attach`](#) ([`\_Safe\_sequence\_base`](#) \*`__seq`, bool `__constant`)
- void [`\_M\_attach\_single`](#) ([`\_Safe\_sequence\_base`](#) \*`__seq`, bool `__constant`) throw ()
- void [`\_M\_attach\_single`](#) ([`\_Safe\_sequence\_base`](#) \*`__seq`)
- bool [`\_M\_attached\_to`](#) (const [`\_Safe\_sequence\_base`](#) \*`__seq`) const
- bool [`\_M\_before\_dereferenceable`](#) () const
- bool [`\_M\_can\_advance`](#) (const `difference_type` &`__n`) const
- bool [`\_M\_can\_compare`](#) (const [`\_Safe\_iterator\_base`](#) &`__x`) const throw ()

- `bool _M_decrementable () const`
- `bool _M_dereferenceable () const`
- `void _M_detach ()`
- `void _M_detach_single () throw ()`
- `const _Sequence * _M_get_sequence () const`
- `bool _M_incrementable () const`
- `void _M_invalidate ()`
- `bool _M_is_before_begin () const`
- `bool _M_is_begin () const`
- `bool _M_is_end () const`
- `void _M_reset () throw ()`
- `bool _M_singular () const throw ()`
- `void _M_unlink () throw ()`
- `template<typename _Other >`  
`bool _M_valid_range (const _Safe_iterator<_Other, _Sequence> &__rhs)`  
`const`
- `_Iterator base () const`
- `operator _Iterator () const`
- `reference operator* () const`
- `_Safe_iterator operator+ (const difference_type &__n) const`
- `_Safe_iterator & operator++ ()`
- `_Safe_iterator operator++ (int)`
- `_Safe_iterator & operator+= (const difference_type &__n)`
- `_Safe_iterator operator- (const difference_type &__n) const`
- `_Safe_iterator & operator-- ()`
- `_Safe_iterator operator-- (int)`
- `_Safe_iterator & operator-= (const difference_type &__n)`
- `pointer operator-> () const`
- `_Safe_iterator & operator= (const _Safe_iterator &__x)`
- `reference operator[] (const difference_type &__n) const`

#### Static Public Member Functions

- `template<typename _Iterator1, typename _Iterator2 >`  
`static std::pair< difference_type, _Distance_precision > _M_get_distance`  
`(const _Iterator1 &__lhs, const _Iterator2 &__rhs)`
- `template<typename _Iterator1, typename _Iterator2 >`  
`static std::pair< difference_type, _Distance_precision > _M_get_distance`  
`(const _Iterator1 &__lhs, const _Iterator2 &__rhs, std::forward_iterator_tag)`
- `template<typename _Iterator1, typename _Iterator2 >`  
`static std::pair< difference_type, _Distance_precision > _M_get_distance`  
`(const _Iterator1 &__lhs, const _Iterator2 &__rhs, std::random_access_`  
`iterator_tag)`

### Public Attributes

- `_Safe_iterator_base * _M_next`
- `_Safe_iterator_base * _M_prior`
- `_Safe_sequence_base * _M_sequence`
- `unsigned int _M_version`

### Protected Member Functions

- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`

#### 5.80.1 Detailed Description

`template<typename _Iterator, typename _Sequence> class __gnu_debug::_Safe_iterator<_Iterator, _Sequence>`

Safe iterator wrapper. The class template `_Safe_iterator` is a wrapper around an iterator that tracks the iterator's movement among sequences and checks that operations performed on the "safe" iterator are legal. In addition to the basic iterator operations (which are validated, and then passed to the underlying iterator), `_Safe_iterator` has member functions for iterator invalidation, attaching/detaching the iterator from sequences, and querying the iterator's state.

Definition at line 77 of file `safe_iterator.h`.

#### 5.80.2 Constructor & Destructor Documentation

**5.80.2.1** `template<typename _Iterator, typename _Sequence>  
__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_Safe_iterator  
( ) [inline]`

### Postcondition

the iterator is singular and unattached

Definition at line 113 of file `safe_iterator.h`.

**5.80.2.2** `template<typename _Iterator, typename _Sequence>  
__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_Safe_iterator  
( const _Iterator & __i, const _Sequence * __seq ) [inline]`

Safe iterator construction from an unsafe iterator and its sequence.

**Precondition**

seq is not NULL

**Postcondition**

this is not singular

Definition at line 122 of file safe\_iterator.h.

References \_GLIBCXX\_DEBUG\_VERIFY, and \_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_singular().

**5.80.2.3** `template<typename _Iterator, typename _Sequence>  
__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_Safe_iterator  
( const _Safe_iterator<_Iterator, _Sequence> & __x ) [inline]`

Copy construction.

Definition at line 133 of file safe\_iterator.h.

References \_GLIBCXX\_DEBUG\_VERIFY, and \_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_singular().

**5.80.2.4** `template<typename _Iterator, typename _Sequence>  
template<typename _MutableIterator> __gnu_debug::_ -  
Safe_iterator<_Iterator, _Sequence>::_Safe_iterator  
( const _Safe_iterator<_MutableIterator, typename  
__gnu_cxx::__enable_if<(std::__are_same<_MutableIterator,  
typename _Sequence::iterator::iterator_type>::_value), _Sequence  
>::_type> & __x ) [inline]`

Converting constructor from a mutable iterator to a constant iterator.

Definition at line 150 of file safe\_iterator.h.

References \_GLIBCXX\_DEBUG\_VERIFY.

### 5.80.3 Member Function Documentation

**5.80.3.1** `template<typename _Iterator, typename _Sequence> void  
__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_attach (  
_Safe_sequence_base * __seq ) [inline]`

Attach iterator to the given sequence.

Definition at line 335 of file `safe_iterator.h`.

Referenced by `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::operator=()`.

**5.80.3.2** `void __gnu_debug::_Safe_iterator_base::_M_attach (  
_Safe_sequence_base * __seq, bool __constant ) [inherited]`

Attaches this iterator to the given sequence, detaching it from whatever sequence it was attached to originally. If the new sequence is the NULL pointer, the iterator is left unattached.

Referenced by `__gnu_debug::_Safe_iterator_base::_Safe_iterator_base()`.

**5.80.3.3** `template<typename _Iterator, typename _Sequence>  
void __gnu_debug::_Safe_iterator<_Iterator, _Sequence  
>::_M_attach_single ( _Safe_sequence_base * __seq ) [inline]`

Likewise, but not thread-safe.

Definition at line 342 of file `safe_iterator.h`.

**5.80.3.4** `void __gnu_debug::_Safe_iterator_base::_M_attach_single  
( _Safe_sequence_base * __seq, bool __constant ) throw ()  
[inherited]`

Likewise, but not thread-safe.

**5.80.3.5** `bool __gnu_debug::_Safe_iterator_base::_M_attached_to ( const  
_Safe_sequence_base * __seq ) const [inline, inherited]`

Determines if we are attached to the given sequence.

Definition at line 130 of file `safe_base.h`.

References `__gnu_debug::_Safe_iterator_base::_M_sequence`.

**5.80.3.6** `template<typename _Iterator, typename _Sequence>  
bool __gnu_debug::_Safe_iterator<_Iterator, _Sequence  
>::_M_before_dereferenceable ( ) const [inline]`

Is the iterator before a dereferenceable one?

Definition at line 354 of file `safe_iterator.h`.

References `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_dereferenceable()`, and `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_incrementable()`.

**5.80.3.7** `bool __gnu_debug::_Safe_iterator_base::_M_can_compare ( const  
_Safe_iterator_base & __x ) const throw () [inherited]`

Can we compare this iterator to the given iterator `__x`? Returns true if both iterators are nonsingular and reference the same sequence.

**5.80.3.8** `template<typename _Iterator, typename _Sequence>  
bool __gnu_debug::_Safe_iterator<_Iterator, _Sequence  
>::_M_dereferenceable ( ) const [inline]`

Is the iterator dereferenceable?

Definition at line 349 of file `safe_iterator.h`.

References `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_is_before_begin()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_is_end()`, and `__gnu_debug::_Safe_iterator_base::_M_singular()`.

Referenced by `__gnu_debug::_check_dereferenceable()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_before_dereferenceable()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::operator*()`, and `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::operator->()`.

**5.80.3.9** `void __gnu_debug::_Safe_iterator_base::_M_detach ( )  
[inherited]`

Detach the iterator for whatever sequence it is attached to, if any.

**5.80.3.10** void \_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_detach\_single ( )  
throw () [inherited]

Likewise, but not thread-safe.

Referenced by \_\_gnu\_debug::\_Safe\_sequence< \_Sequence >::\_M\_transfer\_from\_if().

**5.80.3.11** template<typename \_Iterator, typename \_Sequence>  
template<typename \_Iterator1, typename \_Iterator2 >  
static std::pair<difference\_type, Distance\_precision>  
\_\_gnu\_debug::\_Safe\_iterator< \_Iterator, \_Sequence  
>::\_M\_get\_distance ( const \_Iterator1 & \_\_lhs, const \_Iterator2 &  
\_\_rhs ) [inline, static]

Determine the distance between two iterators with some known precision.

Definition at line 388 of file safe\_iterator.h.

**5.80.3.12** \_\_gnu\_cxx::\_\_mutex& \_\_gnu\_debug::\_Safe\_iterator\_  
base::\_M\_get\_mutex ( ) throw () [protected,  
inherited]

For use in [\\_Safe\\_iterator](#).

**5.80.3.13** template<typename \_Iterator, typename \_Sequence>  
bool \_\_gnu\_debug::\_Safe\_iterator< \_Iterator, \_Sequence  
>::\_M\_incrementable ( ) const [inline]

Is the iterator incrementable?

Definition at line 362 of file safe\_iterator.h.

References \_\_gnu\_debug::\_Safe\_iterator< \_Iterator, \_Sequence >::\_M\_is\_end(), and  
\_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_singular().

Referenced by \_\_gnu\_debug::\_Safe\_iterator< \_Iterator, \_Sequence >::\_M\_  
before\_dereferenceable(), and \_\_gnu\_debug::\_Safe\_iterator< \_Iterator, \_Sequence  
>::operator++().

**5.80.3.14** void \_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_invalidate ( )  
[inline, inherited]

Invalidate the iterator, making it singular.



Definition at line 143 of file safe\_base.h.

References `__gnu_debug::_Safe_iterator_base::_M_version`.

**5.80.3.15** `template<typename _Iterator, typename _Sequence>`  
`bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence`  
`>::_M_is_before_begin ( ) const [inline]`

Is this iterator equal to the sequence's `before_begin()` iterator if /// any?

Definition at line 417 of file safe\_iterator.h.

References `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::base()`.

Referenced by `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_dereferenceable()`.

**5.80.3.16** `template<typename _Iterator, typename _Sequence> bool`  
`__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_is_begin`  
`( ) const [inline]`

Is this iterator equal to the sequence's `begin()` iterator?

Definition at line 408 of file safe\_iterator.h.

References `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::base()`.

**5.80.3.17** `template<typename _Iterator, typename _Sequence> bool`  
`__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_is_end (`  
`) const [inline]`

Is this iterator equal to the sequence's `end()` iterator?

Definition at line 412 of file safe\_iterator.h.

References `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::base()`.

Referenced by `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_dereferenceable()`, and `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_incrementable()`.

**5.80.3.18 void \_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_reset ( ) throw ()**  
**[inherited]**

Reset all member variables

**5.80.3.19 bool \_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_singular ( ) const**  
**throw () [inherited]**

Is this iterator singular?

Referenced by \_\_gnu\_debug::\_check\_singular(), \_\_gnu\_debug::\_check\_singular\_aux(), \_\_gnu\_debug::\_Safe\_iterator<\_Iterator, \_Sequence>::\_M\_dereferenceable(), \_\_gnu\_debug::\_Safe\_iterator<\_Iterator, \_Sequence>::\_M\_incrementable(), \_\_gnu\_debug::\_Safe\_iterator<\_Iterator, \_Sequence>::\_Safe\_iterator(), and \_\_gnu\_debug::\_Safe\_iterator<\_Iterator, \_Sequence>::operator=().

**5.80.3.20 void \_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_unlink ( ) throw ()**  
**[inline, inherited]**

Unlink itself

Definition at line 152 of file safe\_base.h.

References \_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_next, and \_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_prior.

**5.80.3.21 template<typename \_Iterator, typename \_Sequence> \_Iterator**  
**\_\_gnu\_debug::\_Safe\_iterator<\_Iterator, \_Sequence>::base ( )**  
**const [inline]**

Return the underlying iterator.

Definition at line 325 of file safe\_iterator.h.

Referenced by \_\_gnu\_debug::\_Safe\_iterator<\_Iterator, \_Sequence>::\_M\_is\_before\_begin(), \_\_gnu\_debug::\_Safe\_iterator<\_Iterator, \_Sequence>::\_M\_is\_begin(), and \_\_gnu\_debug::\_Safe\_iterator<\_Iterator, \_Sequence>::\_M\_is\_end().

**5.80.3.22 template<typename \_Iterator, typename \_Sequence>**  
**\_\_gnu\_debug::\_Safe\_iterator<\_Iterator, \_Sequence>::operator**  
**\_Iterator ( ) const [inline]**

## 5.80 `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>` Class Template Reference 1088

---

Conversion to underlying non-debug iterator to allow better interaction with non-debug containers.

Definition at line 331 of file `safe_iterator.h`.

**5.80.3.23** `template<typename _Iterator, typename _Sequence> reference  
__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::operator* ( ) const [inline]`

Iterator dereference.

### Precondition

iterator is dereferenceable

Definition at line 189 of file `safe_iterator.h`.

References `_GLIBCXX_DEBUG_VERIFY`, and `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_dereferenceable()`.

**5.80.3.24** `template<typename _Iterator, typename _Sequence> _Safe_iterator  
__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::operator++ ( int ) [inline]`

Iterator postincrement.

### Precondition

iterator is incrementable

Definition at line 232 of file `safe_iterator.h`.

References `_GLIBCXX_DEBUG_VERIFY`, and `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_incrementable()`.

**5.80.3.25** `template<typename _Iterator, typename _Sequence>  
_Safe_iterator& __gnu_debug::_Safe_iterator<_Iterator, _Sequence>::operator++ ( ) [inline]`

Iterator preincrement.

**Precondition**

iterator is incrementable

Definition at line 218 of file safe\_iterator.h.

References `_GLIBCXX_DEBUG_VERIFY`, and `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_incrementable()`.

**5.80.3.26** `template<typename _Iterator, typename _Sequence> _Safe_iterator  
__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator-- (   
int ) [inline]`

Iterator postdecrement.

**Precondition**

iterator is decrementable

Definition at line 262 of file safe\_iterator.h.

References `_GLIBCXX_DEBUG_VERIFY`.

**5.80.3.27** `template<typename _Iterator, typename _Sequence>  
_Safe_iterator& __gnu_debug::_Safe_iterator< _Iterator, _Sequence  
>::operator-- ( ) [inline]`

Iterator predecrement.

**Precondition**

iterator is decrementable

Definition at line 248 of file safe\_iterator.h.

References `_GLIBCXX_DEBUG_VERIFY`.

**5.80.3.28** `template<typename _Iterator, typename _Sequence> pointer  
__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator-> (   
) const [inline]`

Iterator dereference.

**Precondition**

iterator is dereferenceable

**Todo**

Make this correct w.r.t. iterators that return proxies

Use [addressof\(\)](#) instead of & operator

Definition at line 204 of file safe\_iterator.h.

References `_GLIBCXX_DEBUG_VERIFY`, and `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_dereferenceable()`.

```
5.80.3.29 template<typename _Iterator, typename _Sequence>
 _Safe_iterator& __gnu_debug::_Safe_iterator<_Iterator, _Sequence>::operator= (const _Safe_iterator<_Iterator, _Sequence> & __x
) [inline]
```

Copy assignment.

Definition at line 170 of file safe\_iterator.h.

References `_GLIBCXX_DEBUG_VERIFY`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_attach()`, `__gnu_debug::_Safe_iterator_base::_M_sequence`, and `__gnu_debug::_Safe_iterator_base::_M_singular()`.

**5.80.4 Member Data Documentation**

**5.80.4.1** `_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_next`  
**[inherited]**

Pointer to the next iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 73 of file safe\_base.h.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_from_if()`, and `__gnu_debug::_Safe_iterator_base::_M_unlink()`.

**5.80.4.2** `_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_prior`  
**[inherited]**

Pointer to the previous iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 69 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if()`, and `__gnu_debug::_Safe_iterator_base::_M_unlink()`.

#### 5.80.4.3 `_Safe_sequence_base* __gnu_debug::_Safe_iterator_base::_M_sequence` [inherited]

The sequence this iterator references; may be NULL to indicate a singular iterator.

Definition at line 56 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_iterator_base::_M_attached_to()`, `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if()`, `__gnu_debug::_Safe_iterator_base::_Safe_iterator_base()`, and `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator=()`.

#### 5.80.4.4 `unsigned int __gnu_debug::_Safe_iterator_base::_M_version` [inherited]

The version number of this iterator.

The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by `_M_sequence` for the iterator to be non-singular.

Definition at line 65 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_iterator_base::_M_invalidate()`.

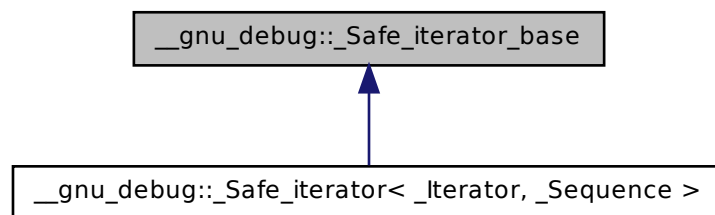
The documentation for this class was generated from the following files:

- [safe\\_iterator.h](#)
- [safe\\_iterator.tcc](#)

## 5.81 `__gnu_debug::_Safe_iterator_base` Class Reference

Basic functionality for a *safe* iterator.

Inheritance diagram for \_\_gnu\_debug::\_Safe\_iterator\_base:



### Public Member Functions

- void [\\_M\\_attach](#) ([\\_Safe\\_sequence\\_base](#) \*\_\_seq, bool \_\_constant)
- void [\\_M\\_attach\\_single](#) ([\\_Safe\\_sequence\\_base](#) \*\_\_seq, bool \_\_constant) throw ()
- bool [\\_M\\_attached\\_to](#) (const [\\_Safe\\_sequence\\_base](#) \*\_\_seq) const
- bool [\\_M\\_can\\_compare](#) (const [\\_Safe\\_iterator\\_base](#) &\_\_x) const throw ()
- void [\\_M\\_detach](#) ()
- void [\\_M\\_detach\\_single](#) () throw ()
- void [\\_M\\_invalidate](#) ()
- void [\\_M\\_reset](#) () throw ()
- bool [\\_M\\_singular](#) () const throw ()
- void [\\_M\\_unlink](#) () throw ()

### Public Attributes

- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_next](#)
- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_prior](#)
- [\\_Safe\\_sequence\\_base](#) \* [\\_M\\_sequence](#)
- unsigned int [\\_M\\_version](#)

### Protected Member Functions

- [\\_Safe\\_iterator\\_base](#) ()
- [\\_Safe\\_iterator\\_base](#) (const [\\_Safe\\_sequence\\_base](#) \*\_\_seq, bool \_\_constant)
- [\\_Safe\\_iterator\\_base](#) (const [\\_Safe\\_iterator\\_base](#) &)

- `_Safe_iterator_base` (const `_Safe_iterator_base` &\_\_x, bool \_\_constant)
- `__gnu_cxx::__mutex` & `_M_get_mutex` () throw ()
- `_Safe_iterator_base` & `operator=` (const `_Safe_iterator_base` &)

### 5.81.1 Detailed Description

Basic functionality for a *safe* iterator. The `_Safe_iterator_base` base class implements the functionality of a safe iterator that is not specific to a particular iterator type. It contains a pointer back to the sequence it references along with iterator version information and pointers to form a doubly-linked list of iterators referenced by the container.

This class must not perform any operations that can throw an exception, or the exception guarantees of derived iterators will be broken.

Definition at line 51 of file `safe_base.h`.

### 5.81.2 Constructor & Destructor Documentation

#### 5.81.2.1 `__gnu_debug::_Safe_iterator_base::_Safe_iterator_base ( )` [`inline`, `protected`]

Initializes the iterator and makes it singular.

Definition at line 77 of file `safe_base.h`.

#### 5.81.2.2 `__gnu_debug::_Safe_iterator_base::_Safe_iterator_base ( const _Safe_sequence_base * __seq, bool __constant )` [`inline`, `protected`]

Initialize the iterator to reference the sequence pointed to by `__seq`. `__constant` is true when we are initializing a constant iterator, and false if it is a mutable iterator. Note that `__seq` may be NULL, in which case the iterator will be singular. Otherwise, the iterator will reference `__seq` and be nonsingular.

Definition at line 88 of file `safe_base.h`.

References `_M_attach()`.

#### 5.81.2.3 `__gnu_debug::_Safe_iterator_base::_Safe_iterator_base ( const _Safe_iterator_base & __x, bool __constant )` [`inline`, `protected`]

Initializes the iterator to reference the same sequence that `__x` does. `__constant` is true if this is a constant iterator, and false if it is mutable.

Definition at line 95 of file `safe_base.h`.



References `_M_attach()`, and `_M_sequence`.

### 5.81.3 Member Function Documentation

#### 5.81.3.1 void \_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_attach ( \_Safe\_sequence\_base \* \_\_seq, bool \_\_constant )

Attaches this iterator to the given sequence, detaching it from whatever sequence it was attached to originally. If the new sequence is the NULL pointer, the iterator is left unattached.

Referenced by `_Safe_iterator_base()`.

#### 5.81.3.2 void \_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_attach\_single ( \_Safe\_sequence\_base \* \_\_seq, bool \_\_constant ) throw ()

Likewise, but not thread-safe.

#### 5.81.3.3 bool \_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_attached\_to ( const \_Safe\_sequence\_base \* \_\_seq ) const [inline]

Determines if we are attached to the given sequence.

Definition at line 130 of file `safe_base.h`.

References `_M_sequence`.

#### 5.81.3.4 bool \_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_can\_compare ( const \_Safe\_iterator\_base & \_\_x ) const throw ()

Can we compare this iterator to the given iterator `__x`? Returns true if both iterators are nonsingular and reference the same sequence.

#### 5.81.3.5 void \_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_detach ( )

Detach the iterator for whatever sequence it is attached to, if any.

#### 5.81.3.6 void \_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_detach\_single ( ) throw ()

Likewise, but not thread-safe.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_from_if()`.

### 5.81.3.7 `__gnu_cxx::mutex& __gnu_debug::_Safe_iterator_base::_M_get_mutex ( ) throw () [protected]`

For use in [\\_Safe\\_iterator](#).

### 5.81.3.8 `void __gnu_debug::_Safe_iterator_base::_M_invalidate ( ) [inline]`

Invalidate the iterator, making it singular.

Definition at line 143 of file `safe_base.h`.

References `_M_version`.

### 5.81.3.9 `void __gnu_debug::_Safe_iterator_base::_M_reset ( ) throw ()`

Reset all member variables

### 5.81.3.10 `bool __gnu_debug::_Safe_iterator_base::_M_singular ( ) const throw ()`

Is this iterator singular?

Referenced by `__gnu_debug::__check_singular()`, `__gnu_debug::__check_singular_aux()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_dereferenceable()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_incrementable()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_Safe_iterator()`, and `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::operator=()`.

### 5.81.3.11 `void __gnu_debug::_Safe_iterator_base::_M_unlink ( ) throw () [inline]`

Unlink itself

Definition at line 152 of file `safe_base.h`.

References `_M_next`, and `_M_prior`.

## 5.81.4 Member Data Documentation

### 5.81.4.1 `_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_next`

Pointer to the next iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

## 5.82 `__gnu_debug::_Safe_sequence< _Sequence >` Class Template Reference

Definition at line 73 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if()`, and `_M_unlink()`.

### 5.81.4.2 `_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_prior`

Pointer to the previous iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 69 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if()`, and `_M_unlink()`.

### 5.81.4.3 `_Safe_sequence_base* __gnu_debug::_Safe_iterator_base::_M_sequence`

The sequence this iterator references; may be `NULL` to indicate a singular iterator.

Definition at line 56 of file `safe_base.h`.

Referenced by `_M_attached_to()`, `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if()`, `_Safe_iterator_base()`, and `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator=()`.

### 5.81.4.4 `unsigned int __gnu_debug::_Safe_iterator_base::_M_version`

The version number of this iterator.

The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by `_M_sequence` for the iterator to be non-singular.

Definition at line 65 of file `safe_base.h`.

Referenced by `_M_invalidate()`.

The documentation for this class was generated from the following file:

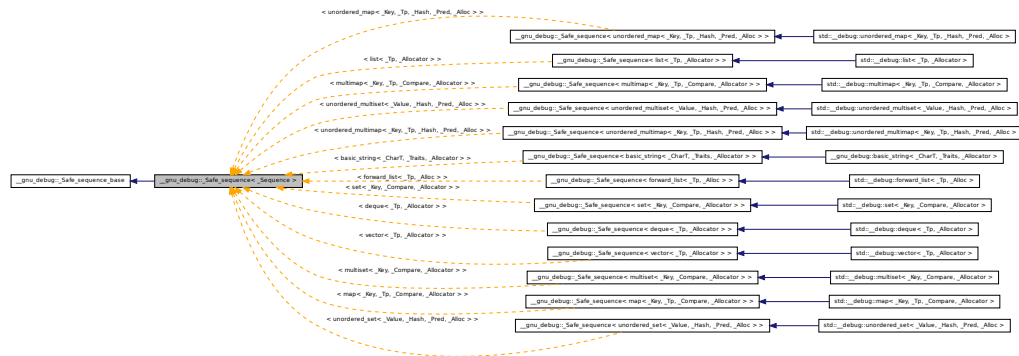
- [safe\\_base.h](#)

## 5.82 `__gnu_debug::_Safe_sequence< _Sequence >` Class Template Reference

Base class for constructing a *safe* sequence type that tracks iterators that reference it.

## 5.82 `__gnu_debug::_Safe_sequence<_Sequence>` Class Template Reference

Inheritance diagram for `__gnu_debug::_Safe_sequence<_Sequence>`:



### Public Member Functions

- `void _M_attach (_Safe_iterator_base * __it, bool __constant)`
- `void _M_attach_single (_Safe_iterator_base * __it, bool __constant) throw ()`
- `void _M_detach (_Safe_iterator_base * __it)`
- `void _M_detach_single (_Safe_iterator_base * __it) throw ()`
- `void _M_invalidate_all () const`
- `template<typename _Predicate>`  
`void _M_invalidate_if (_Predicate __pred)`
- `template<typename _Predicate>`  
`void _M_transfer_from_if (_Safe_sequence & __from, _Predicate __pred)`

### Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

### Protected Member Functions

- `void _M_detach_all ()`
- `void _M_detach_singular ()`
- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`
- `void _M_revalidate_singular ()`
- `void _M_swap (_Safe_sequence_base & __x)`

### 5.82.1 Detailed Description

**template<typename \_Sequence> class \_\_gnu\_debug::\_Safe\_sequence< \_Sequence >**

Base class for constructing a *safe* sequence type that tracks iterators that reference it. The class template `_Safe_sequence` simplifies the construction of *safe* sequences that track the iterators that reference the sequence, so that the iterators are notified of changes in the sequence that may affect their operation, e.g., if the container invalidates its iterators or is destructed. This class template may only be used by deriving from it and passing the name of the derived class as its template parameter via the curiously recurring template pattern. The derived class must have `iterator` and types that are instantiations of class template `_Safe_iterator` for this sequence. Iterators will then be tracked automatically.

Definition at line 112 of file `safe_sequence.h`.

### 5.82.2 Member Function Documentation

**5.82.2.1 void \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_attach (**  
    **`_Safe_iterator_base * __it, bool __constant` ) `[inherited]`**

Attach an iterator to this sequence.

**5.82.2.2 void \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_attach\_single**  
    **( `_Safe_iterator_base * __it, bool __constant` ) throw ()**  
    **`[inherited]`**

Likewise but not thread safe.

**5.82.2.3 void \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_detach (**  
    **`_Safe_iterator_base * __it` ) `[inherited]`**

Detach an iterator from this sequence

**5.82.2.4 void \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_detach\_all ( )**  
    **`[protected, inherited]`**

Detach all iterators, leaving them singular.

Referenced by `__gnu_debug::_Safe_sequence_base::~~_Safe_sequence_base()`.

## 5.82 `__gnu_debug::_Safe_sequence<_Sequence>` Class Template Reference 1099

**5.82.2.5** `void __gnu_debug::_Safe_sequence_base::_M_detach_single (   
_Safe_iterator_base * __it ) throw () [inherited]`

Likewise but not thread safe.

**5.82.2.6** `void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( )   
[protected, inherited]`

Detach all singular iterators.

### Postcondition

for all iterators *i* attached to this sequence, `i->_M_version == _M_version`.

**5.82.2.7** `__gnu_cxx::mutex& __gnu_debug::_Safe_sequence_  
base::_M_get_mutex ( ) throw () [protected,  
inherited]`

For use in [\\_Safe\\_sequence](#).

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if()`,  
and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_from_if()`.

**5.82.2.8** `void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const   
[inline, inherited]`

Invalidates all iterators.

Definition at line 234 of file `safe_base.h`.

References `__gnu_debug::_Safe_sequence_base::_M_version`.

**5.82.2.9** `template<typename _Sequence> template<typename _Predicate>   
void __gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if (   
_Predicate __pred )`

Invalidates all iterators *x* that reference  
this sequence, are not singular, and for which `pred(x)` returns `true`. `pred` will be  
invoked with the normal iterators nested in the safe ones.

Definition at line 38 of file `safe_sequence.tcc`.

References `__gnu_debug::_Safe_sequence_base::_M_const_iterators`, `__gnu_  
debug::_Safe_sequence_base::_M_get_mutex()`, and `__gnu_debug::_Safe_sequence_  
base::_M_iterators`.

## 5.82 `__gnu_debug::_Safe_sequence<_Sequence>` Class Template Reference

**5.82.2.10** `void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( ) [protected, inherited]`

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

**5.82.2.11** `void __gnu_debug::_Safe_sequence_base::_M_swap ( _Safe_sequence_base & __x ) [protected, inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

**5.82.2.12** `template<typename _Sequence> template<typename _Predicate> void __gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_from_if ( _Safe_sequence<_Sequence> & __from, _Predicate __pred )`

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `pred(x)` returns `true`. `pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 69 of file `safe_sequence.tcc`.

References `__gnu_debug::_Safe_sequence_base::_M_const_iterators`, `__gnu_debug::_Safe_iterator_base::_M_detach_single()`, `__gnu_debug::_Safe_sequence_base::_M_get_mutex()`, `__gnu_debug::_Safe_sequence_base::_M_iterators`, `__gnu_debug::_Safe_iterator_base::_M_next`, `__gnu_debug::_Safe_iterator_base::_M_prior`, `__gnu_debug::_Safe_iterator_base::_M_sequence`, and `__gnu_debug::_Safe_sequence_base::_M_version`.

### 5.82.3 Member Data Documentation

**5.82.3.1** `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]`

The list of constant iterators that reference this container.

Definition at line 185 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if()`, and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_from_if()`.

### 5.82.3.2 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators` [inherited]

The list of mutable iterators that reference this container.

Definition at line 182 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if()`, and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_from_if()`.

### 5.82.3.3 `unsigned int __gnu_debug::_Safe_sequence_base::_M_version` [mutable, inherited]

The container version number. This number may never be 0.

Definition at line 188 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`, and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_from_if()`.

The documentation for this class was generated from the following files:

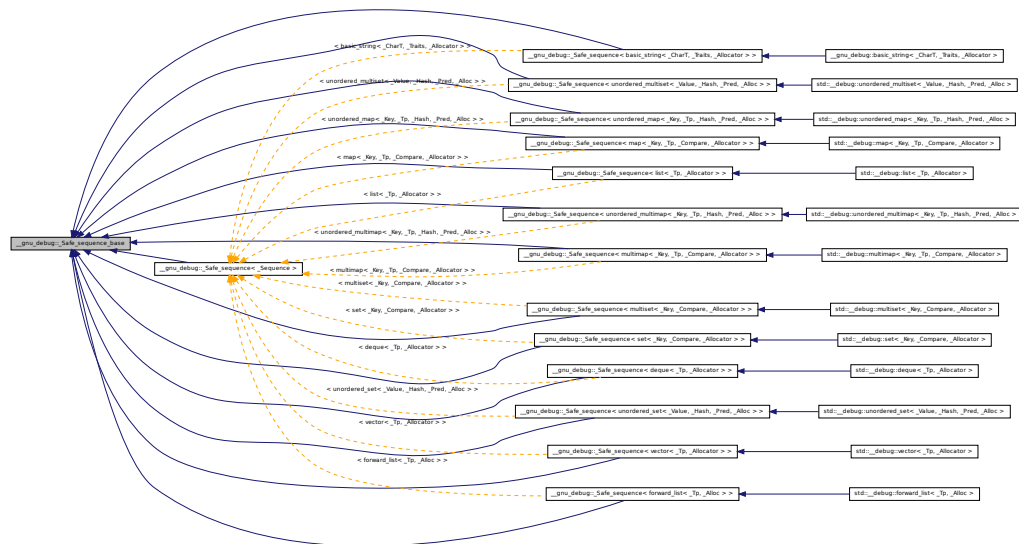
- [safe\\_sequence.h](#)
- [safe\\_sequence.tcc](#)

## 5.83 `__gnu_debug::_Safe_sequence_base` Class Reference

Base class that supports tracking of iterators that reference a sequence.



Inheritance diagram for \_\_gnu\_debug::Safe\_sequence\_base:



## Public Member Functions

- void [\\_M\\_attach](#) ([\\_Safe\\_iterator\\_base](#) \* \_\_it, bool \_\_constant)
- void [\\_M\\_attach\\_single](#) ([\\_Safe\\_iterator\\_base](#) \* \_\_it, bool \_\_constant) throw ()
- void [\\_M\\_detach](#) ([\\_Safe\\_iterator\\_base](#) \* \_\_it)
- void [\\_M\\_detach\\_single](#) ([\\_Safe\\_iterator\\_base](#) \* \_\_it) throw ()
- void [\\_M\\_invalidate\\_all](#) () const

## Public Attributes

- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_const\\_iterators](#)
- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_iterators](#)
- unsigned int [\\_M\\_version](#)

## Protected Member Functions

- [~\\_Safe\\_sequence\\_base](#) ()
- void [\\_M\\_detach\\_all](#) ()
- void [\\_M\\_detach\\_singular](#) ()
- [\\_\\_gnu\\_cxx::\\_\\_mutex](#) & [\\_M\\_get\\_mutex](#) () throw ()
- void [\\_M\\_revalidate\\_singular](#) ()
- void [\\_M\\_swap](#) ([\\_Safe\\_sequence\\_base](#) & \_\_x)

### 5.83.1 Detailed Description

Base class that supports tracking of iterators that reference a sequence. The `_Safe_sequence_base` class provides basic support for tracking iterators into a sequence. Sequences that track iterators must derived from `_Safe_sequence_base` publicly, so that safe iterators (which inherit `_Safe_iterator_base`) can attach to them. This class contains two linked lists of iterators, one for constant iterators and one for mutable iterators, and a version number that allows very fast invalidation of all iterators that reference the container.

This class must ensure that no operation on it may throw an exception, otherwise *safe* sequences may fail to provide the exception-safety guarantees required by the C++ standard.

Definition at line 178 of file `safe_base.h`.

### 5.83.2 Constructor & Destructor Documentation

#### 5.83.2.1 `__gnu_debug::_Safe_sequence_base::~~Safe_sequence_base ( )` [inline, protected]

Notify all iterators that reference this sequence that the sequence is being destroyed.

Definition at line 198 of file `safe_base.h`.

References `_M_detach_all()`.

### 5.83.3 Member Function Documentation

#### 5.83.3.1 `void __gnu_debug::_Safe_sequence_base::_M_attach (` `_Safe_iterator_base * __it, bool __constant )`

Attach an iterator to this sequence.

#### 5.83.3.2 `void __gnu_debug::_Safe_sequence_base::_M_attach_single (` `_Safe_iterator_base * __it, bool __constant ) throw ()`

Likewise but not thread safe.

#### 5.83.3.3 `void __gnu_debug::_Safe_sequence_base::_M_detach (` `_Safe_iterator_base * __it )`

Detach an iterator from this sequence

**5.83.3.4** `void __gnu_debug::_Safe_sequence_base::_M_detach_all ( )`  
**[protected]**

Detach all iterators, leaving them singular.

Referenced by `~_Safe_sequence_base()`.

**5.83.3.5** `void __gnu_debug::_Safe_sequence_base::_M_detach_single (`  
`_Safe_iterator_base * __it ) throw ()`

Likewise but not thread safe.

**5.83.3.6** `void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( )`  
**[protected]**

Detach all singular iterators.

#### Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

**5.83.3.7** `__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_-`  
`mutex ( ) throw ()` **[protected]**

For use in [\\_Safe\\_sequence](#).

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`,  
and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if()`.

**5.83.3.8** `void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const`  
**[inline]**

Invalidates all iterators.

Definition at line 234 of file `safe_base.h`.

References `_M_version`.

**5.83.3.9** `void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( )`  
**[protected]**

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

#### 5.83.3.10 `void __gnu_debug::_Safe_sequence_base::_M_swap (` `_Safe_sequence_base & __x ) [protected]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

### 5.83.4 Member Data Documentation

#### 5.83.4.1 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_` `const_iterators`

The list of constant iterators that reference this container.

Definition at line 185 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if()`.

#### 5.83.4.2 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_` `iterators`

The list of mutable iterators that reference this container.

Definition at line 182 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if()`.

#### 5.83.4.3 `unsigned int __gnu_debug::_Safe_sequence_base::_M_version` `[mutable]`

The container version number. This number may never be 0.

Definition at line 188 of file `safe_base.h`.

Referenced by `_M_invalidate_all()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if()`.

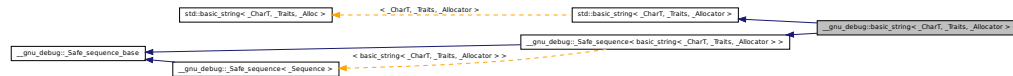
The documentation for this class was generated from the following file:

- [safe\\_base.h](#)

## 5.84 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference

Class `std::basic_string` with safety/checking/debug instrumentation.

Inheritance diagram for `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>`:



### Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::_Safe_iterator<typename _Base::const_iterator, basic_string>` **const\_iterator**
- typedef `__gnu_cxx::__normal_iterator<const_pointer, basic_string>` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_CharT_alloc_type::const_pointer` **const\_pointer**
- typedef `_CharT_alloc_type::const_reference` **const\_reference**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator<const_iterator>` **const\_reverse\_iterator**
- typedef `std::reverse_iterator<const_iterator>` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `_CharT_alloc_type::difference_type` **difference\_type**
- typedef `__gnu_cxx::__normal_iterator<pointer, basic_string>` **iterator**
- typedef `__gnu_debug::_Safe_iterator<typename _Base::iterator, basic_string>` **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_CharT_alloc_type::pointer` **pointer**
- typedef `_CharT_alloc_type::reference` **reference**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator<iterator>` **reverse\_iterator**
- typedef `std::reverse_iterator<iterator>` **reverse\_iterator**
- typedef `_CharT_alloc_type::size_type` **size\_type**
- typedef `_Base::size_type` **size\_type**
- typedef `_Traits` **traits\_type**
- typedef `_Traits` **traits\_type**
- typedef `_Traits::char_type` **value\_type**
- typedef `_Traits::char_type` **value\_type**

## Public Member Functions

- **basic\_string** (const \_Allocator &\_\_a=\_Allocator())
- **basic\_string** (const \_Base &\_\_base)
- **basic\_string** (const [basic\\_string](#) &\_\_str, size\_type \_\_pos, size\_type \_\_n=[Base::npos](#), const \_Allocator &\_\_a=\_Allocator())
- template<typename \_InputIterator >  
**basic\_string** (\_InputIterator \_\_begin, \_InputIterator \_\_end, const \_Allocator &\_\_a=\_Allocator())
- **basic\_string** ([basic\\_string](#) &&\_\_str)
- **basic\_string** (const \_CharT \*\_\_s, size\_type \_\_n, const \_Allocator &\_\_a=\_Allocator())
- **basic\_string** ([std::initializer\\_list](#)< \_CharT > \_\_l, const \_Allocator &\_\_a=\_Allocator())
- **basic\_string** (const [basic\\_string](#) &\_\_str)
- **basic\_string** (const \_CharT \*\_\_s, const \_Allocator &\_\_a=\_Allocator())
- **basic\_string** (size\_type \_\_n, \_CharT \_\_c, const \_Allocator &\_\_a=\_Allocator())
  
- void [\\_M\\_attach](#) (\_Safe\_iterator\_base \*\_\_it, bool \_\_constant)
- void [\\_M\\_attach\\_single](#) (\_Safe\_iterator\_base \*\_\_it, bool \_\_constant) throw ()
- [\\_Base](#) & [\\_M\\_base](#) ()
- const [\\_Base](#) & [\\_M\\_base](#) () const
- void [\\_M\\_detach](#) (\_Safe\_iterator\_base \*\_\_it)
- void [\\_M\\_detach\\_single](#) (\_Safe\_iterator\_base \*\_\_it) throw ()
- void [\\_M\\_invalidate\\_all](#) () const
- void [\\_M\\_invalidate\\_if](#) (\_Predicate \_\_pred)
- void [\\_M\\_transfer\\_from\\_if](#) (\_Safe\_sequence &\_\_from, \_Predicate \_\_pred)
- [basic\\_string](#) & **append** (const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & **append** (const [basic\\_string](#) &\_\_str, size\_type \_\_pos, size\_type \_\_n)
- [basic\\_string](#) & **append** (const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & **append** (const \_CharT \*\_\_s, size\_type \_\_n)
- [basic\\_string](#) & **append** (const [basic\\_string](#) &\_\_str, size\_type \_\_pos, size\_type \_\_n)
- [basic\\_string](#) & **append** (const \_CharT \*\_\_s, size\_type \_\_n)
- [basic\\_string](#) & **append** (const \_CharT \*\_\_s)
- [basic\\_string](#) & **append** (size\_type \_\_n, \_CharT \_\_c)
- [basic\\_string](#) & **append** (const \_CharT \*\_\_s)
- [basic\\_string](#) & **append** ([initializer\\_list](#)< \_CharT > \_\_l)
- [basic\\_string](#) & **append** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- [basic\\_string](#) & **append** (size\_type \_\_n, \_CharT \_\_c)
- template<typename \_InputIterator >  
[basic\\_string](#) & **append** (\_InputIterator \_\_first, \_InputIterator \_\_last)

- `basic_string` & `assign` (const `basic_string` &\_\_str)
- `basic_string` & `assign` (`basic_string` &&\_\_str)
- `basic_string` & `assign` (const `basic_string` &\_\_str, size\_type \_\_pos, size\_type \_\_n)
- `basic_string` & `assign` (const \_CharT \*\_\_s, size\_type \_\_n)
- `basic_string` & `assign` (const \_CharT \*\_\_s)
- `basic_string` & `assign` (size\_type \_\_n, \_CharT \_\_c)
- `basic_string` & `assign` (\_InputIterator \_\_first, \_InputIterator \_\_last)
- `basic_string` & `assign` (initializer\_list<\_CharT> \_\_l)
- `basic_string` & `assign` (const `basic_string` &\_\_x)
- `basic_string` & `assign` (`basic_string` &&\_\_x)
- `basic_string` & `assign` (const `basic_string` &\_\_str, size\_type \_\_pos, size\_type \_\_n)
- `basic_string` & `assign` (const \_CharT \*\_\_s, size\_type \_\_n)
- `basic_string` & `assign` (const \_CharT \*\_\_s)
- `basic_string` & `assign` (size\_type \_\_n, \_CharT \_\_c)
- template<typename \_InputIterator>  
  `basic_string` & `assign` (\_InputIterator \_\_first, \_InputIterator \_\_last)
- `basic_string` & `assign` (std::initializer\_list<\_CharT> \_\_l)
- const\_reference at (size\_type \_\_n) const
- reference at (size\_type \_\_n)
- reference back ()
- const\_reference back () const
- iterator begin ()
- const\_iterator begin () const
- iterator begin ()
- const\_iterator begin () const
- const \_CharT \* c\_str () const
- const \_CharT \* c\_str () const
- size\_type capacity () const
- const\_iterator cbegin () const
- const\_iterator cend () const
- void clear ()
- void clear ()
- int compare (const `basic_string` &\_\_str) const
- int compare (size\_type \_\_pos1, size\_type \_\_n1, const `basic_string` &\_\_str) const
- int compare (size\_type \_\_pos1, size\_type \_\_n1, const `basic_string` &\_\_str, size\_type \_\_pos2, size\_type \_\_n2) const
- int compare (size\_type \_\_pos1, size\_type \_\_n1, const \_CharT \*\_\_s) const
- int compare (const \_CharT \*\_\_s) const
- int compare (size\_type \_\_pos1, size\_type \_\_n1, const \_CharT \*\_\_s, size\_type \_\_n2) const

- `int compare` (const `basic_string` &\_\_str) const
- `int compare` (size\_type \_\_pos, size\_type \_\_n, const `basic_string` &\_\_str) const
- `int compare` (const `_CharT` \*\_\_s) const
- `int compare` (size\_type \_\_pos, size\_type \_\_n1, const `_CharT` \*\_\_s, size\_type \_\_n2) const
- `int compare` (size\_type \_\_pos1, size\_type \_\_n1, const `basic_string` &\_\_str, size\_type \_\_pos2, size\_type \_\_n2) const
- `int compare` (size\_type \_\_pos, size\_type \_\_n1, const `_CharT` \*\_\_s) const
- size\_type `copy` (`_CharT` \*\_\_s, size\_type \_\_n, size\_type \_\_pos=0) const
- size\_type `copy` (`_CharT` \*\_\_s, size\_type \_\_n, size\_type \_\_pos=0) const
- `const_reverse_iterator crbegin` () const
- `const_reverse_iterator crend` () const
- const `_CharT` \* `data` () const
- const `_CharT` \* `data` () const
- bool `empty` () const
- `iterator end` ()
- `const_iterator end` () const
- `iterator end` ()
- `const_iterator end` () const
- `basic_string` & `erase` (size\_type \_\_pos=0, size\_type \_\_n=`npos`)
- `iterator erase` (`iterator` \_\_position)
- `iterator erase` (`iterator` \_\_first, `iterator` \_\_last)
- `basic_string` & `erase` (size\_type \_\_pos=0, size\_type \_\_n=`_Base::npos`)
- `iterator erase` (`iterator` \_\_position)
- `iterator erase` (`iterator` \_\_first, `iterator` \_\_last)
- size\_type `find` (const `basic_string` &\_\_str, size\_type \_\_pos=0) const
- size\_type `find` (const `basic_string` &\_\_str, size\_type \_\_pos=0) const
- size\_type `find` (const `_CharT` \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const
- size\_type `find` (const `_CharT` \*\_\_s, size\_type \_\_pos=0) const
- size\_type `find` (`_CharT` \_\_c, size\_type \_\_pos=0) const
- size\_type `find` (const `_CharT` \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const
- size\_type `find` (const `_CharT` \*\_\_s, size\_type \_\_pos=0) const
- size\_type `find` (`_CharT` \_\_c, size\_type \_\_pos=0) const
- size\_type `find_first_not_of` (`_CharT` \_\_c, size\_type \_\_pos=0) const
- size\_type `find_first_not_of` (const `_CharT` \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const
- size\_type `find_first_not_of` (`_CharT` \_\_c, size\_type \_\_pos=0) const
- size\_type `find_first_not_of` (const `_CharT` \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const
- size\_type `find_first_not_of` (const `_CharT` \*\_\_s, size\_type \_\_pos=0) const
- size\_type `find_first_not_of` (const `basic_string` &\_\_str, size\_type \_\_pos=0) const
- size\_type `find_first_not_of` (const `basic_string` &\_\_str, size\_type \_\_pos=0) const



- `size_type find_first_not_of` (const `_CharT` \*`__s`, `size_type` `__pos`=0) const
- `size_type find_first_of` (const `_CharT` \*`__s`, `size_type` `__pos`, `size_type` `__n`) const
- `size_type find_first_of` (const `_CharT` \*`__s`, `size_type` `__pos`=0) const
- `size_type find_first_of` (`_CharT` `__c`, `size_type` `__pos`=0) const
- `size_type find_first_of` (const `basic_string` &`__str`, `size_type` `__pos`=0) const
- `size_type find_first_of` (const `_CharT` \*`__s`, `size_type` `__pos`, `size_type` `__n`) const
- `size_type find_first_of` (const `_CharT` \*`__s`, `size_type` `__pos`=0) const
- `size_type find_first_of` (`_CharT` `__c`, `size_type` `__pos`=0) const
- `size_type find_first_of` (const `basic_string` &`__str`, `size_type` `__pos`=0) const
- `size_type find_last_not_of` (`_CharT` `__c`, `size_type` `__pos`=`Base::npos`) const
- `size_type find_last_not_of` (const `_CharT` \*`__s`, `size_type` `__pos`, `size_type` `__n`) const
- `size_type find_last_not_of` (const `_CharT` \*`__s`, `size_type` `__pos`=`Base::npos`) const
- `size_type find_last_not_of` (const `basic_string` &`__str`, `size_type` `__pos`=`Base::npos`) const
- `size_type find_last_not_of` (const `basic_string` &`__str`, `size_type` `__pos`=`npos`) const
- `size_type find_last_not_of` (const `_CharT` \*`__s`, `size_type` `__pos`=`npos`) const
- `size_type find_last_not_of` (const `_CharT` \*`__s`, `size_type` `__pos`, `size_type` `__n`) const
- `size_type find_last_not_of` (`_CharT` `__c`, `size_type` `__pos`=`npos`) const
- `size_type find_last_of` (`_CharT` `__c`, `size_type` `__pos`=`npos`) const
- `size_type find_last_of` (const `_CharT` \*`__s`, `size_type` `__pos`, `size_type` `__n`) const
- `size_type find_last_of` (const `basic_string` &`__str`, `size_type` `__pos`=`Base::npos`) const
- `size_type find_last_of` (const `_CharT` \*`__s`, `size_type` `__pos`, `size_type` `__n`) const
- `size_type find_last_of` (const `_CharT` \*`__s`, `size_type` `__pos`=`Base::npos`) const
- `size_type find_last_of` (`_CharT` `__c`, `size_type` `__pos`=`Base::npos`) const
- `size_type find_last_of` (const `basic_string` &`__str`, `size_type` `__pos`=`npos`) const
- `size_type find_last_of` (const `_CharT` \*`__s`, `size_type` `__pos`=`npos`) const
- reference `front` ()
- `const_reference front` () const
- `allocator_type get_allocator` () const
- void `insert` (iterator `__p`, `std::initializer_list`< `_CharT` > `__l`)
- `basic_string` & `insert` (`size_type` `__pos`, `size_type` `__n`, `_CharT` `__c`)
- void `insert` (iterator `__p`, `_InputIterator` `__beg`, `_InputIterator` `__end`)
- void `insert` (iterator `__p`, `initializer_list`< `_CharT` > `__l`)

- `basic_string & insert` (size\_type \_\_pos1, const `basic_string` &\_\_str)
- `basic_string & insert` (size\_type \_\_pos, const `_CharT *`\_\_s, size\_type \_\_n)
- `basic_string & insert` (size\_type \_\_pos, const `_CharT *`\_\_s)
- `basic_string & insert` (size\_type \_\_pos, size\_type \_\_n, `_CharT` \_\_c)
- `basic_string & insert` (size\_type \_\_pos1, const `basic_string` &\_\_str, size\_type \_\_pos2, size\_type \_\_n)
- `iterator insert` (iterator \_\_p, `_CharT` \_\_c)
- `basic_string & insert` (size\_type \_\_pos1, const `basic_string` &\_\_str)
- `iterator insert` (iterator \_\_p, `_CharT` \_\_c)
- `basic_string & insert` (size\_type \_\_pos1, const `basic_string` &\_\_str, size\_type \_\_pos2, size\_type \_\_n)
- `basic_string & insert` (size\_type \_\_pos, const `_CharT *`\_\_s)
- `void insert` (iterator \_\_p, size\_type \_\_n, `_CharT` \_\_c)
- `void insert` (iterator \_\_p, size\_type \_\_n, `_CharT` \_\_c)
- `basic_string & insert` (size\_type \_\_pos, const `_CharT *`\_\_s, size\_type \_\_n)
- `template<typename _InputIterator>`  
  `void insert` (iterator \_\_p, `_InputIterator` \_\_first, `_InputIterator` \_\_last)
- size\_type `length` () const
- size\_type `max_size` () const
- `basic_string & operator+=` (`_CharT` \_\_c)
- `basic_string & operator+=` (`std::initializer_list`< `_CharT` > \_\_l)
- `basic_string & operator+=` (const `basic_string` &\_\_str)
- `basic_string & operator+=` (const `_CharT *`\_\_s)
- `basic_string & operator+=` (`initializer_list`< `_CharT` > \_\_l)
- `basic_string & operator+=` (`_CharT` \_\_c)
- `basic_string & operator+=` (const `basic_string` &\_\_str)
- `basic_string & operator+=` (const `_CharT *`\_\_s)
- `basic_string & operator=` (`std::initializer_list`< `_CharT` > \_\_l)
- `basic_string & operator=` (const `_CharT *`\_\_s)
- `basic_string & operator=` (const `basic_string` &\_\_str)
- `basic_string & operator=` (`basic_string` &&\_\_str)
- `basic_string & operator=` (`_CharT` \_\_c)
- const\_reference `operator[]` (size\_type \_\_pos) const
- reference `operator[]` (size\_type \_\_pos)
- const\_reference `operator[]` (size\_type \_\_pos) const
- reference `operator[]` (size\_type \_\_pos)
- `void push_back` (`_CharT` \_\_c)
- `void push_back` (`_CharT` \_\_c)
- `reverse_iterator rbegin` ()
- const `reverse_iterator rbegin` () const
- `reverse_iterator rbegin` ()
- const `reverse_iterator rbegin` () const
- const `reverse_iterator rend` () const

- `const_reverse_iterator` `rend` () const
- `reverse_iterator` `rend` ()
- `reverse_iterator` `rend` ()
- `basic_string` & `replace` (`iterator` \_\_i1, `iterator` \_\_i2, `size_type` \_\_n, `_CharT` \_\_c)
- `basic_string` & `replace` (`iterator` \_\_i1, `iterator` \_\_i2, `iterator` \_\_k1, `iterator` \_\_k2)
- `basic_string` & `replace` (`size_type` \_\_pos, `size_type` \_\_n, const `basic_string` &\_\_str)
- `basic_string` & `replace` (`iterator` \_\_i1, `iterator` \_\_i2, const `_CharT` \*\_\_k1, const `_CharT` \*\_\_k2)
- `basic_string` & `replace` (`iterator` \_\_i1, `iterator` \_\_i2, const `_CharT` \*\_\_s, `size_type` \_\_n)
- `basic_string` & `replace` (`iterator` \_\_i1, `iterator` \_\_i2, `_CharT` \*\_\_k1, `_CharT` \*\_\_k2)
- `basic_string` & `replace` (`iterator` \_\_i1, `iterator` \_\_i2, `size_type` \_\_n, `_CharT` \_\_c)
- `basic_string` & `replace` (`size_type` \_\_pos, `size_type` \_\_n1, const `_CharT` \*\_\_s, `size_type` \_\_n2)
- `basic_string` & `replace` (`size_type` \_\_pos1, `size_type` \_\_n1, const `basic_string` &\_\_str)
- `basic_string` & `replace` (`iterator` \_\_i1, `iterator` \_\_i2, const `basic_string` &\_\_str)
- `basic_string` & `replace` (`iterator` \_\_i1, `iterator` \_\_i2, const `_CharT` \*\_\_s, `size_type` \_\_n)
- `basic_string` & `replace` (`iterator` \_\_i1, `iterator` \_\_i2, const `_CharT` \*\_\_s)
- `basic_string` & `replace` (`size_type` \_\_pos, `size_type` \_\_n1, const `_CharT` \*\_\_s)
- `basic_string` & `replace` (`iterator` \_\_i1, `iterator` \_\_i2, const `_CharT` \*\_\_s)
- `basic_string` & `replace` (`iterator` \_\_i1, `iterator` \_\_i2, `initializer_list`< `_CharT` > \_\_l)
- `basic_string` & `replace` (`size_type` \_\_pos, `size_type` \_\_n1, `size_type` \_\_n2, `_CharT` \_\_c)
- `basic_string` & `replace` (`iterator` \_\_i1, `iterator` \_\_i2, `_InputIterator` \_\_k1, `_InputIterator` \_\_k2)
- `template`<`typename` `_InputIterator` >  
  `basic_string` & `replace` (`iterator` \_\_i1, `iterator` \_\_i2, `_InputIterator` \_\_j1, `_InputIterator` \_\_j2)
- `basic_string` & `replace` (`size_type` \_\_pos1, `size_type` \_\_n1, const `basic_string` &\_\_str, `size_type` \_\_pos2, `size_type` \_\_n2)
- `basic_string` & `replace` (`size_type` \_\_pos, `size_type` \_\_n1, const `_CharT` \*\_\_s)
- `basic_string` & `replace` (`size_type` \_\_pos, `size_type` \_\_n1, const `_CharT` \*\_\_s, `size_type` \_\_n2)
- `basic_string` & `replace` (`size_type` \_\_pos1, `size_type` \_\_n1, const `basic_string` &\_\_str, `size_type` \_\_pos2, `size_type` \_\_n2)
- `basic_string` & `replace` (`iterator` \_\_i1, `iterator` \_\_i2, `std::initializer_list`< `_CharT` > \_\_l)
- `basic_string` & `replace` (`size_type` \_\_pos, `size_type` \_\_n1, `size_type` \_\_n2, `_CharT` \_\_c)

- `basic_string` & `replace` (`iterator` \_\_i1, `iterator` \_\_i2, const `basic_string` &\_\_str)
- void `reserve` (`size_type` \_\_res\_arg=0)
- void `resize` (`size_type` \_\_n)
- void `resize` (`size_type` \_\_n, `_CharT` \_\_c)
- void `resize` (`size_type` \_\_n)
- void `resize` (`size_type` \_\_n, `_CharT` \_\_c)
- `size_type` `rfind` (const `_CharT` \*\_\_s, `size_type` \_\_pos, `size_type` \_\_n) const
- `size_type` `rfind` (`_CharT` \_\_c, `size_type` \_\_pos=`npos`) const
- `size_type` `rfind` (const `basic_string` &\_\_str, `size_type` \_\_pos=`_Base::npos`) const
- `size_type` `rfind` (const `_CharT` \*\_\_s, `size_type` \_\_pos=`npos`) const
- `size_type` `rfind` (const `basic_string` &\_\_str, `size_type` \_\_pos=`npos`) const
- `size_type` `rfind` (`_CharT` \_\_c, `size_type` \_\_pos=`_Base::npos`) const
- `size_type` `rfind` (const `_CharT` \*\_\_s, `size_type` \_\_pos, `size_type` \_\_n) const
- `size_type` `rfind` (const `_CharT` \*\_\_s, `size_type` \_\_pos=`_Base::npos`) const
- void `shrink_to_fit` ()
- `size_type` `size` () const
- `basic_string` `substr` (`size_type` \_\_pos=0, `size_type` \_\_n=`_Base::npos`) const
- `basic_string` `substr` (`size_type` \_\_pos=0, `size_type` \_\_n=`npos`) const
- void `swap` (`basic_string` &\_\_s)
- void `swap` (`basic_string`< `_CharT`, `_Traits`, `_Allocator` > &\_\_x)

#### Public Attributes

- `_Safe_iterator_base` \* `_M_const_iterators`
- `_Safe_iterator_base` \* `_M_iterators`
- unsigned int `_M_version`

#### Static Public Attributes

- static const `size_type` `npos`

#### Protected Member Functions

- void `_M_detach_all` ()
- void `_M_detach_singular` ()
- `__gnu_cxx::__mutex` & `_M_get_mutex` () throw ()
- void `_M_revalidate_singular` ()
- void `_M_swap` (`_Safe_sequence_base` &\_\_x)

### 5.84.1 Detailed Description

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>,
typename _Allocator = std::allocator<_CharT>> class __gnu_debug::basic_
string<_CharT, _Traits, _Allocator>
```

Class `std::basic_string` with safety/checking/debug instrumentation.

Definition at line 42 of file `debug/string`.

### 5.84.2 Member Function Documentation

5.84.2.1 `void __gnu_debug::Safe_sequence_base::M_attach (`  
`_Safe_iterator_base * __it, bool __constant ) [inherited]`

Attach an iterator to this sequence.

5.84.2.2 `void __gnu_debug::Safe_sequence_base::M_attach_single`  
`( _Safe_iterator_base * __it, bool __constant ) throw ()`  
`[inherited]`

Likewise but not thread safe.

5.84.2.3 `void __gnu_debug::Safe_sequence_base::M_detach (`  
`_Safe_iterator_base * __it ) [inherited]`

Detach an iterator from this sequence

5.84.2.4 `void __gnu_debug::Safe_sequence_base::M_detach_all ( )`  
`[protected, inherited]`

Detach all iterators, leaving them singular.

Referenced by `__gnu_debug::Safe_sequence_base::~~Safe_sequence_base()`.

5.84.2.5 `void __gnu_debug::Safe_sequence_base::M_detach_single (`  
`_Safe_iterator_base * __it ) throw () [inherited]`

Likewise but not thread safe.

5.84.2.6 `void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( )`  
`[protected, inherited]`

Detach all singular iterators.

**Postcondition**

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

5.84.2.7 `__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_-`  
`base::_M_get_mutex ( ) throw ( ) [protected,`  
`inherited]`

For use in [\\_Safe\\_sequence](#).

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if()`,  
and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_from_if()`.

5.84.2.8 `void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const`  
`[inline, inherited]`

Invalidates all iterators.

Definition at line 234 of file `safe_base.h`.

References `__gnu_debug::_Safe_sequence_base::_M_version`.

5.84.2.9 `void __gnu_debug::_Safe_sequence<basic_string<_CharT,`  
`_Traits, _Allocator>>::_M_invalidate_if ( _Predicate __pred )`  
`[inherited]`

Invalidates all iterators `x` that reference  
this sequence, are not singular, and for which `pred(x)` returns `true`. `pred` will be  
invoked with the normal iterators nested in the safe ones.

5.84.2.10 `void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular (`  
`) [protected, inherited]`

Revalidates all attached singular iterators. This method may be used to validate  
iterators that were invalidated before (but for some reason, such as an exception, need  
to become valid again).

5.84.2.11 `void __gnu_debug::_Safe_sequence_base::_M_swap (`  
`_Safe_sequence_base & __x ) [protected, inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

5.84.2.12 `void __gnu_debug::_Safe_sequence< basic_string<_CharT,`  
`_Traits, _Allocator> >::_M_transfer_from_if ( _Safe_sequence<`  
`basic_string<_CharT, _Traits, _Allocator> > & __from,`  
`_Predicate __pred ) [inherited]`

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `pred(x)` returns `true`. `pred` will be invoked with the normal iterators nested in the safe ones.

5.84.2.13 `basic_string& std::basic_string<_CharT, _Traits, _Allocator`  
`>::append ( const basic_string<_CharT, _Traits, _Allocator> &`  
`__str ) [inherited]`

Append a string to this string.

#### Parameters

*str* The string to append.

#### Returns

Reference to this string.

5.84.2.14 `basic_string& std::basic_string<_CharT, _Traits, _Allocator`  
`>::append ( const basic_string<_CharT, _Traits, _Allocator> &`  
`__str, size_type __pos, size_type __n ) [inherited]`

Append a substring.

#### Parameters

*str* The string to append.

*pos* Index of the first character of *str* to append.

*n* The number of characters to append.

#### Returns

Reference to this string.

#### Exceptions

[\*std::out\\_of\\_range\*](#) if *pos* is not a valid index.

This function appends *n* characters from *str* starting at *pos* to this string. If *n* is larger than the number of available characters in *str*, the remainder of *str* is appended.

**5.84.2.15** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::append ( const _CharT * __s, size_type __n )` [inherited]

Append a C substring.

#### Parameters

*s* The C string to append.

*n* The number of characters to append.

#### Returns

Reference to this string.

**5.84.2.16** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::append ( const _CharT * __s )` [inline, inherited]

Append a C string.

#### Parameters

*s* The C string to append.

#### Returns

Reference to this string.

Definition at line 997 of file `basic_string.h`.



**5.84.2.17** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::append ( size_type __n, _CharT __c ) [inherited]`

Append multiple characters.

**Parameters**

- n* The number of characters to append.
- c* The character to use.

**Returns**

Reference to this string.

Appends *n* copies of *c* to this string.

**5.84.2.18** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::append ( initializer_list<_CharT> __l ) [inline, inherited]`

Append an `initializer_list` of characters.

**Parameters**

- l* The `initializer_list` of characters to append.

**Returns**

Reference to this string.

Definition at line 1021 of file `basic_string.h`.

**5.84.2.19** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::append ( _InputIterator __first, _InputIterator __last ) [inline, inherited]`

Append a range of characters.

**Parameters**

- first* Iterator referencing the first character to append.

*last* Iterator marking the end of the range.

#### Returns

Reference to this string.

Appends characters in the range [first,last) to this string.

Definition at line 1035 of file `basic_string.h`.

**5.84.2.20** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::assign ( const basic_string<_CharT, _Traits, _Allocator> & __str ) [inherited]`

Set value to contents of another string.

#### Parameters

*str* Source string to use.

#### Returns

Reference to this string.

**5.84.2.21** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::assign ( basic_string<_CharT, _Traits, _Allocator> && __str ) [inline, inherited]`

Set value to contents of another string.

#### Parameters

*str* Source string to use.

#### Returns

Reference to this string.

This function sets this string to the exact contents of *str*. *str* is a valid, but unspecified string.

Definition at line 1070 of file `basic_string.h`.

5.84.2.22 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::assign ( const basic_string<_CharT, _Traits, _Allocator> & __str, size_type __pos, size_type __n ) [inline, inherited]`

Set value to a substring of a string.

#### Parameters

*str* The string to use.  
*pos* Index of the first character of *str*.  
*n* Number of characters to use.

#### Returns

Reference to this string.

#### Exceptions

[\*std::out\\_of\\_range\*](#) if *pos* is not a valid index.

This function sets this string to the substring of *str* consisting of *n* characters at *pos*. If *n* is larger than the number of available characters in *str*, the remainder of *str* is used.

Definition at line 1090 of file `basic_string.h`.

5.84.2.23 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::assign ( const _CharT* __s, size_type __n ) [inherited]`

Set value to a C substring.

#### Parameters

*s* The C string to use.  
*n* Number of characters to use.

#### Returns

Reference to this string.

This function sets the value of this string to the first *n* characters of *s*. If *n* is larger than the number of available characters in *s*, the remainder of *s* is used.

**5.84.2.24** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::assign ( const _CharT * __s ) [inline, inherited]`

Set value to contents of a C string.

**Parameters**

*s* The C string to use.

**Returns**

Reference to this string.

This function sets the value of this string to the value of *s*. The data is copied, so there is no dependence on *s* once the function returns.

Definition at line 1118 of file `basic_string.h`.

**5.84.2.25** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::assign ( size_type __n, _CharT __c ) [inline, inherited]`

Set value to multiple characters.

**Parameters**

*n* Length of the resulting string.

*c* The character to use.

**Returns**

Reference to this string.

This function sets the value of this string to *n* copies of character *c*.

Definition at line 1134 of file `basic_string.h`.

**5.84.2.26** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::assign ( _InputIterator __first, _InputIterator __last ) [inline, inherited]`

Set value to a range of characters.

**Parameters**

*first* Iterator referencing the first character to append.

*last* Iterator marking the end of the range.

**Returns**

Reference to this string.

Sets value of string to characters in the range [first,last).

Definition at line 1147 of file basic\_string.h.

**5.84.2.27 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::assign ( initializer_list<_CharT> __l ) [inline, inherited]`**

Set value to an initializer\_list of characters.

**Parameters**

*l* The initializer\_list of characters to assign.

**Returns**

Reference to this string.

Definition at line 1157 of file basic\_string.h.

**5.84.2.28 `const_reference std::basic_string<_CharT, _Traits, _Allocator>::at ( size_type __n ) const [inline, inherited]`**

Provides access to the data contained in the string.

**Parameters**

*n* The index of the character to access.

**Returns**

Read-only (const) reference to the character.

**Exceptions**

*[std::out\\_of\\_range](#)* If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails.

Definition at line 856 of file `basic_string.h`.

**5.84.2.29** `reference std::basic_string<_CharT, _Traits, _Allocator>::at (`  
`size_type __n ) [inline, inherited]`

Provides access to the data contained in the string.

#### Parameters

*n* The index of the character to access.

#### Returns

Read/write reference to the character.

#### Exceptions

*std::out\_of\_range* If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails. Success results in unsharing the string.

Definition at line 909 of file `basic_string.h`.

**5.84.2.30** `reference std::basic_string<_CharT, _Traits, _Allocator>::back (`  
`) [inline, inherited]`

Returns a read/write reference to the data at the last element of the string.

Definition at line 885 of file `basic_string.h`.

**5.84.2.31** `const_reference std::basic_string<_CharT, _Traits, _Allocator`  
`>::back ( ) const [inline, inherited]`

Returns a read-only (constant) reference to the data at the last element of the string.

Definition at line 893 of file `basic_string.h`.

**5.84.2.32 iterator std::basic\_string< \_CharT, \_Traits, \_Allocator >::begin ( )**  
**[inline, inherited]**

Returns a read/write iterator that points to the first character in the string. Unshares the string.

Definition at line 600 of file basic\_string.h.

**5.84.2.33 const\_iterator std::basic\_string< \_CharT, \_Traits, \_Allocator >::begin ( ) const**  
**[inline, inherited]**

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 611 of file basic\_string.h.

**5.84.2.34 const \_CharT\* std::basic\_string< \_CharT, \_Traits, \_Allocator >::c\_str ( ) const**  
**[inline, inherited]**

Return const pointer to null-terminated contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1766 of file basic\_string.h.

**5.84.2.35 size\_type std::basic\_string< \_CharT, \_Traits, \_Allocator >::capacity ( ) const**  
**[inline, inherited]**

Returns the total number of characters that the string can hold before needing to allocate more memory.

Definition at line 768 of file basic\_string.h.

**5.84.2.36 const\_iterator std::basic\_string< \_CharT, \_Traits, \_Allocator >::cbegin ( ) const**  
**[inline, inherited]**

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 675 of file basic\_string.h.

**5.84.2.37 const\_iterator std::basic\_string< \_CharT, \_Traits, \_Allocator >::cend ( ) const**  
**[inline, inherited]**

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 683 of file `basic_string.h`.

**5.84.2.38** `void std::basic_string<_CharT, _Traits, _Allocator>::clear ( )`  
`[inline, inherited]`

Erases the string, making it empty.

Definition at line 795 of file `basic_string.h`.

**5.84.2.39** `int std::basic_string<_CharT, _Traits, _Allocator>::compare (`  
`const _CharT * __s ) const` `[inherited]`

Compare to a C string.

#### Parameters

*s* C string to compare against.

#### Returns

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Returns an integer  $< 0$  if this string is ordered before *s*,  $0$  if their values are equivalent, or  $> 0$  if this string is ordered after *s*. Determines the effective length *rlen* of the strings to compare as the smallest of `size()` and the length of a string constructed from *s*. The function then compares the two strings by calling `traits::compare(data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**5.84.2.40** `int std::basic_string<_CharT, _Traits, _Allocator>::compare (`  
`const basic_string<_CharT, _Traits, _Allocator> & __str ) const`  
`[inline, inherited]`

Compare to a string.

#### Parameters

*str* String to compare against.

#### Returns

Integer  $< 0$ ,  $0$ , or  $> 0$ .



Returns an integer  $< 0$  if this string is ordered before *str*,  $0$  if their values are equivalent, or  $> 0$  if this string is ordered after *str*. Determines the effective length *rlen* of the strings to compare as the smallest of *size()* and *str.size()*. The function then compares the two strings by calling `traits::compare(data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 2173 of file `basic_string.h`.

**5.84.2.41** `int std::basic_string<_CharT, _Traits, _Allocator>::compare (`  
`size_type __pos, size_type __n, const basic_string<_CharT,`  
`_Traits, _Allocator> & __str ) const [inherited]`

Compare substring to a string.

#### Parameters

*pos* Index of first character of substring.

*n* Number of characters in substring.

*str* String to compare against.

#### Returns

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Form the substring of this string from the *n* characters starting at *pos*. Returns an integer  $< 0$  if the substring is ordered before *str*,  $0$  if their values are equivalent, or  $> 0$  if the substring is ordered after *str*. Determines the effective length *rlen* of the strings to compare as the smallest of the length of the substring and *str.size()*. The function then compares the two strings by calling `traits::compare(substring.data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**5.84.2.42** `int std::basic_string<_CharT, _Traits, _Allocator>::compare (`  
`size_type __pos1, size_type __n1, const basic_string<_CharT,`  
`_Traits, _Allocator> & __str, size_type __pos2, size_type __n2 )`  
`const [inherited]`

Compare substring to a substring.

#### Parameters

*pos1* Index of first character of substring.

*n1* Number of characters in substring.  
*str* String to compare against.  
*pos2* Index of first character of substring of *str*.  
*n2* Number of characters in substring of *str*.

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the *n1* characters starting at *pos1*. Form the substring of *str* from the *n2* characters starting at *pos2*. Returns an integer < 0 if this substring is ordered before the substring of *str*, 0 if their values are equivalent, or > 0 if this substring is ordered after the substring of *str*. Determines the effective length *rlen* of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling `traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**5.84.2.43** `int std::basic_string<_CharT, _Traits, _Allocator>::compare (`  
    `size_type __pos, size_type __n1, const _CharT * __s, size_type`  
    `__n2 ) const [inherited]`

Compare substring against a character array.

**Parameters**

*pos1* Index of first character of substring.  
*n1* Number of characters in substring.  
*s* character array to compare against.  
*n2* Number of characters of *s*.

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the *n1* characters starting at *pos1*. Form a string from the first *n2* characters of *s*. Returns an integer < 0 if this substring is ordered before the string from *s*, 0 if their values are equivalent, or > 0 if this substring is ordered after the string from *s*. Determines the effective length *rlen* of the strings to compare as the smallest of the length of the substring and *n2*. The function then compares the two strings by calling `traits::compare(substring.data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

NB: *s* must have at least *n2* characters, `'\0'` has no special meaning.

**5.84.2.44** `int std::basic_string<_CharT, _Traits, _Allocator>::compare (`  
    `size_type __pos, size_type __n1, const _CharT* __s ) const`  
    `[inherited]`

Compare substring to a C string.

#### Parameters

- pos* Index of first character of substring.
- n1* Number of characters in substring.
- s* C string to compare against.

#### Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *n1* characters starting at *pos*. Returns an integer < 0 if the substring is ordered before *s*, 0 if their values are equivalent, or > 0 if the substring is ordered after *s*. Determines the effective length *rlen* of the strings to compare as the smallest of the length of the substring and the length of a string constructed from *s*. The function then compares the two string by calling `traits::compare(substring.data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**5.84.2.45** `size_type std::basic_string<_CharT, _Traits, _Allocator>::copy`  
    `( _CharT* __s, size_type __n, size_type __pos = 0 ) const`  
    `[inherited]`

Copy substring into C string.

#### Parameters

- s* C string to copy value into.
- n* Number of characters to copy.
- pos* Index of first character to copy.

#### Returns

Number of characters actually copied

#### Exceptions

*std::out\_of\_range* If *pos* > *size()*.

Copies up to *n* characters starting at *pos* into the C string *s*. If *pos* is greater than *size()*, *out\_of\_range* is thrown.

**5.84.2.46 const\_reverse\_iterator std::basic\_string<\_CharT, \_Traits, \_Allocator>::rbegin( ) const [inline, inherited]**

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 692 of file *basic\_string.h*.

**5.84.2.47 const\_reverse\_iterator std::basic\_string<\_CharT, \_Traits, \_Allocator>::crend( ) const [inline, inherited]**

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 701 of file *basic\_string.h*.

**5.84.2.48 const \_CharT\* std::basic\_string<\_CharT, \_Traits, \_Allocator>::data( ) const [inline, inherited]**

Return const pointer to contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1776 of file *basic\_string.h*.

**5.84.2.49 bool std::basic\_string<\_CharT, \_Traits, \_Allocator>::empty( ) const [inline, inherited]**

Returns true if the string is empty. Equivalent to *\*this* == "".

Definition at line 803 of file *basic\_string.h*.

**5.84.2.50 const\_iterator std::basic\_string<\_CharT, \_Traits, \_Allocator>::end( ) const [inline, inherited]**

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 630 of file *basic\_string.h*.

**5.84.2.51** `iterator std::basic_string<_CharT, _Traits, _Allocator>::end ( )`  
`[inline, inherited]`

Returns a read/write iterator that points one past the last character in the string.  
Unshares the string.

Definition at line 619 of file `basic_string.h`.

**5.84.2.52** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::erase ( size_type __pos = 0, size_type __n = npos )`  
`[inline, inherited]`

Remove characters.

**Parameters**

*pos* Index of first character to remove (default 0).

*n* Number of characters to remove (default remainder).

**Returns**

Reference to this string.

**Exceptions**

*std::out\_of\_range* If *pos* is beyond the end of this string.

Removes *n* characters from this string starting at *pos*. The length of the string is reduced by *n*. If there are  $< n$  characters to remove, the remainder of the string is truncated. If *p* is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1347 of file `basic_string.h`.

**5.84.2.53** `iterator std::basic_string<_CharT, _Traits, _Allocator>::erase ( iterator __position )`  
`[inline, inherited]`

Remove one character.

**Parameters**

*position* Iterator referencing the character to remove.

### Returns

iterator referencing same location after removal.

Removes the character at *position* from this string. The value of the string doesn't change if an error is thrown.

Definition at line 1363 of file `basic_string.h`.

**5.84.2.54** `iterator std::basic_string<_CharT, _Traits, _Allocator>::erase (`  
`iterator __first, iterator __last ) [inherited]`

Remove a range of characters.

### Parameters

*first* Iterator referencing the first character to remove.

*last* Iterator referencing the end of the range.

### Returns

Iterator referencing location of first after removal.

Removes the characters in the range [first,last) from this string. The value of the string doesn't change if an error is thrown.

**5.84.2.55** `size_type std::basic_string<_CharT, _Traits, _Allocator>::find`  
`( const _CharT * __s, size_type __pos, size_type __n ) const`  
`[inherited]`

Find position of a C substring.

### Parameters

*s* C string to locate.

*pos* Index of character to search from.

*n* Number of characters from *s* to search for.

### Returns

Index of start of first occurrence.

Starting from *pos*, searches forward for the first *n* characters in *s* within this string. If found, returns the index where it begins. If not found, returns *npos*.

**5.84.2.56** `size_type std::basic_string<_CharT, _Traits, _Allocator>::find (`  
`const basic_string<_CharT, _Traits, _Allocator> & __str, size_type`  
`__pos = 0 ) const [inline, inherited]`

Find position of a string.

#### Parameters

*str* String to locate.

*pos* Index of character to search from (default 0).

#### Returns

Index of start of first occurrence.

Starting from *pos*, searches forward for value of *str* within this string. If found, returns the index where it begins. If not found, returns *npos*.

Definition at line 1811 of file `basic_string.h`.

**5.84.2.57** `size_type std::basic_string<_CharT, _Traits, _Allocator>::find (`  
`_CharT __c, size_type __pos = 0 ) const [inherited]`

Find position of a character.

#### Parameters

*c* Character to locate.

*pos* Index of character to search from (default 0).

#### Returns

Index of first occurrence.

Starting from *pos*, searches forward for *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

**5.84.2.58** `size_type std::basic_string<_CharT, _Traits, _Allocator>::find`  
`( const _CharT * __s, size_type __pos = 0 ) const [inline,`  
`inherited]`

Find position of a C string.

**Parameters**

*s* C string to locate.

*pos* Index of character to search from (default 0).

**Returns**

Index of start of first occurrence.

Starting from *pos*, searches forward for the value of *s* within this string. If found, returns the index where it begins. If not found, returns *npos*.

Definition at line 1825 of file `basic_string.h`.

**5.84.2.59** `size_type std::basic_string<_CharT, _Traits, _Allocator>::find_first_not_of ( const basic_string<_CharT, _Traits, _Allocator> & __str, size_type __pos = 0 ) const` `[inline, inherited]`

Find position of a character not in string.

**Parameters**

*str* String containing characters to avoid.

*pos* Index of character to search from (default 0).

**Returns**

Index of first occurrence.

Starting from *pos*, searches forward for a character not contained in *str* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 2035 of file `basic_string.h`.

**5.84.2.60** `size_type std::basic_string<_CharT, _Traits, _Allocator>::find_first_not_of ( const _CharT* __s, size_type __pos, size_type __n ) const` `[inherited]`

Find position of a character not in C substring.

**Parameters**

*s* C string containing characters to avoid.



*pos* Index of character to search from.  
*n* Number of characters from *s* to consider.

**Returns**

Index of first occurrence.

Starting from *pos*, searches forward for a character not contained in the first *n* characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

**5.84.2.61** `size_type std::basic_string<_CharT, _Traits, _Allocator>::find_first_not_of ( _CharT __c, size_type __pos = 0 ) const`  
`[inherited]`

Find position of a different character.

**Parameters**

*c* Character to avoid.  
*pos* Index of character to search from (default 0).

**Returns**

Index of first occurrence.

Starting from *pos*, searches forward for a character other than *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

**5.84.2.62** `size_type std::basic_string<_CharT, _Traits, _Allocator>::find_first_not_of ( const _CharT * __s, size_type __pos = 0 ) const`  
`[inline, inherited]`

Find position of a character not in C string.

**Parameters**

*s* C string containing characters to avoid.  
*pos* Index of character to search from (default 0).

**Returns**

Index of first occurrence.

Starting from *pos*, searches forward for a character not contained in *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 2064 of file `basic_string.h`.

**5.84.2.63** `size_type std::basic_string<_CharT, _Traits, _Allocator>::find_first_of( _CharT __c, size_type __pos = 0 ) const`  
[inline, inherited]

Find position of a character.

#### Parameters

- c* Character to locate.
- pos* Index of character to search from (default 0).

#### Returns

Index of first occurrence.

Starting from *pos*, searches forward for the character *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Note: equivalent to `find(c, pos)`.

Definition at line 1960 of file `basic_string.h`.

**5.84.2.64** `size_type std::basic_string<_CharT, _Traits, _Allocator>::find_first_of( const _CharT * __s, size_type __pos = 0 ) const`  
[inline, inherited]

Find position of a character of C string.

#### Parameters

- s* String containing characters to locate.
- pos* Index of character to search from (default 0).

#### Returns

Index of first occurrence.

Starting from *pos*, searches forward for one of the characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 1941 of file `basic_string.h`.

**5.84.2.65** `size_type std::basic_string<_CharT, _Traits, _Allocator  
>::find_first_of ( const basic_string<_CharT, _Traits, _Allocator>  
& __str, size_type __pos = 0 ) const [inline, inherited]`

Find position of a character of string.

**Parameters**

*str* String containing characters to locate.  
*pos* Index of character to search from (default 0).

**Returns**

Index of first occurrence.

Starting from *pos*, searches forward for one of the characters of *str* within this string.  
If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 1913 of file `basic_string.h`.

**5.84.2.66** `size_type std::basic_string<_CharT, _Traits, _Allocator  
>::find_first_of ( const _CharT * __s, size_type __pos, size_type  
__n ) const [inherited]`

Find position of a character of C substring.

**Parameters**

*s* String containing characters to locate.  
*pos* Index of character to search from.  
*n* Number of characters from *s* to search for.

**Returns**

Index of first occurrence.

Starting from *pos*, searches forward for one of the first *n* characters of *s* within this  
string. If found, returns the index where it was found. If not found, returns *npos*.

**5.84.2.67** `size_type std::basic_string<_CharT, _Traits, _Allocator  
>::find_last_not_of ( const _CharT * __s, size_type __pos,  
size_type __n ) const [inherited]`

Find last position of a character not in C substring.

**Parameters**

- s* C string containing characters to avoid.
- pos* Index of character to search back from.
- n* Number of characters from *s* to consider.

**Returns**

Index of last occurrence.

Starting from *pos*, searches backward for a character not contained in the first *n* characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

**5.84.2.68** `size_type std::basic_string<_CharT, _Traits, _Allocator  
>::find_last_not_of( _CharT __c, size_type __pos = npo`  
`s ) const`  
`[inherited]`

Find last position of a different character.

**Parameters**

- c* Character to avoid.
- pos* Index of character to search back from (default end).

**Returns**

Index of last occurrence.

Starting from *pos*, searches backward for a character other than *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

**5.84.2.69** `size_type std::basic_string<_CharT, _Traits, _Allocator  
>::find_last_not_of( const _CharT * __s, size_type __pos = npo`  
`st ) const`  
`[inline, inherited]`

Find last position of a character not in C string.

**Parameters**

- s* C string containing characters to avoid.

*pos* Index of character to search back from (default end).

#### Returns

Index of last occurrence.

Starting from *pos*, searches backward for a character not contained in *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 2123 of file `basic_string.h`.

**5.84.2.70** `size_type std::basic_string<_CharT, _Traits, _Allocator>::find_last_not_of ( const basic_string<_CharT, _Traits, _Allocator> & __str, size_type __pos = npos ) const` [`inline`, `inherited`]

Find last position of a character not in string.

#### Parameters

*str* String containing characters to avoid.

*pos* Index of character to search back from (default end).

#### Returns

Index of last occurrence.

Starting from *pos*, searches backward for a character not contained in *str* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 2094 of file `basic_string.h`.

**5.84.2.71** `size_type std::basic_string<_CharT, _Traits, _Allocator>::find_last_of ( const basic_string<_CharT, _Traits, _Allocator> & __str, size_type __pos = npos ) const` [`inline`, `inherited`]

Find last position of a character of string.

#### Parameters

*str* String containing characters to locate.

*pos* Index of character to search back from (default end).

**Returns**

Index of last occurrence.

Starting from *pos*, searches backward for one of the characters of *str* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 1974 of file `basic_string.h`.

**5.84.2.72** `size_type std::basic_string<_CharT, _Traits, _Allocator>::find_last_of( const _CharT * __s, size_type __pos = npos ) const` `[inline, inherited]`

Find last position of a character of C string.

**Parameters**

*s* C string containing characters to locate.

*pos* Index of character to search back from (default end).

**Returns**

Index of last occurrence.

Starting from *pos*, searches backward for one of the characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 2002 of file `basic_string.h`.

**5.84.2.73** `size_type std::basic_string<_CharT, _Traits, _Allocator>::find_last_of( _CharT __c, size_type __pos = npos ) const` `[inline, inherited]`

Find last position of a character.

**Parameters**

*c* Character to locate.

*pos* Index of character to search back from (default end).

**Returns**

Index of last occurrence.

Starting from *pos*, searches backward for *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Note: equivalent to `rfind(c, pos)`.

Definition at line 2021 of file `basic_string.h`.

**5.84.2.74** `size_type std::basic_string<_CharT, _Traits, _Allocator>::find_last_of ( const _CharT * __s, size_type __pos, size_type __n ) const` `[inherited]`

Find last position of a character of C substring.

#### Parameters

*s* C string containing characters to locate.

*pos* Index of character to search back from.

*n* Number of characters from *s* to search for.

#### Returns

Index of last occurrence.

Starting from *pos*, searches backward for one of the first *n* characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

**5.84.2.75** `reference std::basic_string<_CharT, _Traits, _Allocator>::front ( )` `[inline, inherited]`

Returns a read/write reference to the data at the first element of the string.

Definition at line 869 of file `basic_string.h`.

**5.84.2.76** `const_reference std::basic_string<_CharT, _Traits, _Allocator>::front ( ) const` `[inline, inherited]`

Returns a read-only (constant) reference to the data at the first element of the string.

Definition at line 877 of file `basic_string.h`.

**5.84.2.77** `allocator_type std::basic_string<_CharT, _Traits, _Allocator>::get_allocator ( ) const` `[inline, inherited]`

Return copy of allocator used to construct this string.

Definition at line 1783 of file `basic_string.h`.

**5.84.2.78** `void std::basic_string<_CharT, _Traits, _Allocator>::insert  
( iterator __p, size_type __n, _CharT __c ) [inline,  
inherited]`

Insert multiple characters.

#### Parameters

*p* Iterator referencing location in string to insert at.

*n* Number of characters to insert

*c* The character to insert.

#### Exceptions

[\*std::length\\_error\*](#) If new length exceeds `max_size()`.

Inserts *n* copies of character *c* starting at the position referenced by iterator *p*. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1174 of file `basic_string.h`.

**5.84.2.79** `basic_string& std::basic_string<_CharT, _Traits, _Allocator  
>::insert ( size_type __pos1, const basic_string<_CharT,  
_Traits, _Allocator> & __str, size_type __pos2, size_type __n )  
[inline, inherited]`

Insert a substring.

#### Parameters

*pos1* Iterator referencing location in string to insert at.

*str* The string to insert.

*pos2* Start of characters in *str* to insert.

*n* Number of characters to insert.

#### Returns

Reference to this string.



### Exceptions

*`std::length_error`* If new length exceeds `max_size()`.

*`std::out_of_range`* If `pos1 > size()` or `pos2 > str.size()`.

Starting at `pos1`, insert `n` character of `str` beginning with `pos2`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `pos1` is beyond the end of this string or `pos2` is beyond the end of `str`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1242 of file `basic_string.h`.

**5.84.2.80** `void std::basic_string<_CharT, _Traits, _Allocator>::insert  
( iterator __p, initializer_list<_CharT> __l ) [inline,  
inherited]`

Insert an `initializer_list` of characters.

### Parameters

*`p`* Iterator referencing location in string to insert at.

*`l`* The `initializer_list` of characters to insert.

### Exceptions

*`std::length_error`* If new length exceeds `max_size()`.

Definition at line 1201 of file `basic_string.h`.

**5.84.2.81** `iterator std::basic_string<_CharT, _Traits, _Allocator>::insert (  
iterator __p, _CharT __c ) [inline, inherited]`

Insert one character.

### Parameters

*`p`* Iterator referencing position in string to insert at.

*`c`* The character to insert.

### Returns

Iterator referencing newly inserted char.

### Exceptions

*[std::length\\_error](#)* If new length exceeds `max_size()`.

Inserts character *c* at position referenced by *p*. If adding character causes the length to exceed `max_size()`, `length_error` is thrown. If *p* is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1323 of file `basic_string.h`.

**5.84.2.82** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::insert ( size_type __pos1, const basic_string<_CharT, _Traits, _Allocator> & __str ) [inline, inherited]`

Insert value of a string.

### Parameters

*pos1* Iterator referencing location in string to insert at.

*str* The string to insert.

### Returns

Reference to this string.

### Exceptions

*[std::length\\_error](#)* If new length exceeds `max_size()`.

Inserts value of *str* starting at *pos1*. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1220 of file `basic_string.h`.

**5.84.2.83** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::insert ( size_type __pos, const _CharT* __s ) [inline, inherited]`

Insert a C string.

### Parameters

*pos* Iterator referencing location in string to insert at.

*s* The C string to insert.

#### Returns

Reference to this string.

#### Exceptions

*[std::length\\_error](#)* If new length exceeds `max_size()`.

*[std::out\\_of\\_range](#)* If *pos* is beyond the end of this string.

Inserts the first *n* characters of *s* starting at *pos*. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If *pos* is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1283 of file `basic_string.h`.

**5.84.2.84** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::insert( size_type __pos, size_type __n, _CharT __c )`  
`[inline, inherited]`

Insert multiple characters.

#### Parameters

*pos* Index in string to insert at.

*n* Number of characters to insert

*c* The character to insert.

#### Returns

Reference to this string.

#### Exceptions

*[std::length\\_error](#)* If new length exceeds `max_size()`.

*[std::out\\_of\\_range](#)* If *pos* is beyond the end of this string.

Inserts *n* copies of character *c* starting at index *pos*. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If *pos* > `length()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1306 of file `basic_string.h`.

5.84.2.85 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::insert ( size_type __pos, const _CharT* __s, size_type __n )`  
[`inherited`]

Insert a C substring.

**Parameters**

- pos* Iterator referencing location in string to insert at.
- s* The C string to insert.
- n* The number of characters to insert.

**Returns**

Reference to this string.

**Exceptions**

- [\*std::length\\_error\*](#) If new length exceeds `max_size()`.
- [\*std::out\\_of\\_range\*](#) If *pos* is beyond the end of this string.

Inserts the first *n* characters of *s* starting at *pos*. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If *pos* is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

5.84.2.86 `void std::basic_string<_CharT, _Traits, _Allocator>::insert ( iterator __p, _InputIterator __beg, _InputIterator __end )`  
[`inline`, `inherited`]

Insert a range of characters.

**Parameters**

- p* Iterator referencing location in string to insert at.
- beg* Start of range.
- end* End of range.

**Exceptions**

- [\*std::length\\_error\*](#) If new length exceeds `max_size()`.

Inserts characters in range [beg,end). If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1190 of file `basic_string.h`.

**5.84.2.87** `size_type std::basic_string<_CharT, _Traits, _Allocator>::length ( ) const [inline, inherited]`

Returns the number of characters in the string, not including any /// null-termination.

Definition at line 716 of file `basic_string.h`.

**5.84.2.88** `size_type std::basic_string<_CharT, _Traits, _Allocator>::max_size ( ) const [inline, inherited]`

Returns the `size()` of the largest possible string.

Definition at line 721 of file `basic_string.h`.

**5.84.2.89** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::operator+=( const basic_string<_CharT, _Traits, _Allocator> & __str ) [inline, inherited]`

Append a string to this string.

#### Parameters

*str* The string to append.

#### Returns

Reference to this string.

Definition at line 924 of file `basic_string.h`.

**5.84.2.90** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::operator+=( const _CharT * __s ) [inline, inherited]`

Append a C string.

**Parameters**

*s* The C string to append.

**Returns**

Reference to this string.

Definition at line 933 of file `basic_string.h`.

**5.84.2.91** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::operator+=( _CharT __c )` [`inline`, `inherited`]

Append a character.

**Parameters**

*c* The character to append.

**Returns**

Reference to this string.

Definition at line 942 of file `basic_string.h`.

**5.84.2.92** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::operator+=( initializer_list<_CharT> __l )` [`inline`, `inherited`]

Append an `initializer_list` of characters.

**Parameters**

*l* The `initializer_list` of characters to be appended.

**Returns**

Reference to this string.

Definition at line 955 of file `basic_string.h`.

**5.84.2.93** `const_reference std::basic_string<_CharT, _Traits, _Allocator>::operator[]( size_type __pos ) const` `[inline, inherited]`

Subscript access to the data contained in the string.

**Parameters**

*pos* The index of the character to access.

**Returns**

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Definition at line 818 of file `basic_string.h`.

**5.84.2.94** `reference std::basic_string<_CharT, _Traits, _Allocator>::operator[]( size_type __pos )` `[inline, inherited]`

Subscript access to the data contained in the string.

**Parameters**

*pos* The index of the character to access.

**Returns**

Read/write reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.) Unshares the string.

Definition at line 835 of file `basic_string.h`.

**5.84.2.95** `void std::basic_string<_CharT, _Traits, _Allocator>::push_back ( _CharT __c )` `[inline, inherited]`

Append a single character.

**Parameters**

*c* Character to append.

Definition at line 1043 of file `basic_string.h`.

**5.84.2.96** `const_reverse_iterator` `std::basic_string<_CharT, _Traits, _Allocator>::rbegin( )` `const` `[inline, inherited]`

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 648 of file `basic_string.h`.

**5.84.2.97** `reverse_iterator` `std::basic_string<_CharT, _Traits, _Allocator>::rbegin( )` `[inline, inherited]`

Returns a read/write reverse iterator that points to the last character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 639 of file `basic_string.h`.

**5.84.2.98** `const_reverse_iterator` `std::basic_string<_CharT, _Traits, _Allocator>::rend( )` `const` `[inline, inherited]`

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 666 of file `basic_string.h`.

**5.84.2.99** `reverse_iterator` `std::basic_string<_CharT, _Traits, _Allocator>::rend( )` `[inline, inherited]`

Returns a read/write reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 657 of file `basic_string.h`.

**5.84.2.100** `basic_string&` `std::basic_string<_CharT, _Traits, _Allocator>::replace( iterator __i1, iterator __i2, const _CharT* __s )` `[inline, inherited]`

Replace range of characters with C string.



### Parameters

- i1* Iterator referencing start of range to replace.
- i2* Iterator referencing end of range to replace.
- s* C string value to insert.

### Returns

Reference to this string.

### Exceptions

[\*std::length\\_error\*](#) If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, the characters of *s* are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1547 of file `basic_string.h`.

**5.84.2.101** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace ( iterator __i1, iterator __i2, const _CharT * __s, size_type __n ) [inline, inherited]`

Replace range of characters with C substring.

### Parameters

- i1* Iterator referencing start of range to replace.
- i2* Iterator referencing end of range to replace.
- s* C string value to insert.
- n* Number of characters from *s* to insert.

### Returns

Reference to this string.

### Exceptions

[\*std::length\\_error\*](#) If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, the first *n* characters of *s* are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1526 of file `basic_string.h`.

**5.84.2.102** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace ( iterator __i1, iterator __i2, _InputIterator __k1, _InputIterator __k2 ) [inline, inherited]`

Replace range of characters with range.

#### Parameters

- i1* Iterator referencing start of range to replace.
- i2* Iterator referencing end of range to replace.
- k1* Iterator referencing start of range to insert.
- k2* Iterator referencing end of range to insert.

#### Returns

Reference to this string.

#### Exceptions

*std::length\_error* If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, characters in the range `[k1,k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1591 of file `basic_string.h`.

**5.84.2.103** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace ( iterator __i1, iterator __i2, size_type __n, _CharT __c ) [inline, inherited]`

Replace range of characters with multiple characters.

#### Parameters

- i1* Iterator referencing start of range to replace.
- i2* Iterator referencing end of range to replace.
- n* Number of characters to insert.
- c* Character to insert.

#### Returns

Reference to this string.

### Exceptions

*[std::length\\_error](#)* If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, *n* copies of *c* are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1568 of file `basic_string.h`.

**5.84.2.104** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace ( size_type __pos, size_type __n, const basic_string<_CharT, _Traits, _Allocator> & __str ) [inline, inherited]`

Replace characters with value from another string.

### Parameters

*pos* Index of first character to replace.  
*n* Number of characters to be replaced.  
*str* String to insert.

### Returns

Reference to this string.

### Exceptions

*[std::out\\_of\\_range](#)* If *pos* is beyond the end of this string.

*[std::length\\_error](#)* If new length exceeds `max_size()`.

Removes the characters in the range `[pos,pos+n)` from this string. In place, the value of *str* is inserted. If *pos* is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1402 of file `basic_string.h`.

**5.84.2.105** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace ( size_type __pos1, size_type __n1, const basic_string<_CharT, _Traits, _Allocator> & __str, size_type __pos2, size_type __n2 ) [inline, inherited]`

Replace characters with value from another string.

#### Parameters

*pos1* Index of first character to replace.  
*n1* Number of characters to be replaced.  
*str* String to insert.  
*pos2* Index of first character of *str* to use.  
*n2* Number of characters from *str* to use.

#### Returns

Reference to this string.

#### Exceptions

*std::out\_of\_range* If *pos1* > size() or *pos2* > str.size().  
*std::length\_error* If new length exceeds `max_size()`.

Removes the characters in the range [*pos1*, *pos1* + *n*) from this string. In place, the value of *str* is inserted. If *pos* is beyond end of string, *out\_of\_range* is thrown. If the length of the result exceeds `max_size()`, *length\_error* is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1424 of file `basic_string.h`.

**5.84.2.106** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace ( iterator __i1, iterator __i2, const basic_string<_CharT, _Traits, _Allocator> & __str ) [inline, inherited]`

Replace range of characters with string.

#### Parameters

*i1* Iterator referencing start of range to replace.  
*i2* Iterator referencing end of range to replace.  
*str* String value to insert.

#### Returns

Reference to this string.

### Exceptions

*[std::length\\_error](#)* If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, the value of *str* is inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1508 of file `basic_string.h`.

**5.84.2.107** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace ( size_type __pos, size_type __n1, const _CharT* __s ) [inline, inherited]`

Replace characters with value of a C string.

### Parameters

*pos* Index of first character to replace.  
*n1* Number of characters to be replaced.  
*s* C string to insert.

### Returns

Reference to this string.

### Exceptions

*[std::out\\_of\\_range](#)* If *pos* > `size()`.

*[std::length\\_error](#)* If new length exceeds `max_size()`.

Removes the characters in the range `[pos,pos + n1)` from this string. In place, the characters of *s* are inserted. If *pos* is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1467 of file `basic_string.h`.

**5.84.2.108** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace ( size_type __pos, size_type __n1, size_type __n2, _CharT __c ) [inline, inherited]`

Replace characters with multiple characters.

### Parameters

*pos* Index of first character to replace.  
*n1* Number of characters to be replaced.  
*n2* Number of characters to insert.  
*c* Character to insert.

### Returns

Reference to this string.

### Exceptions

*std::out\_of\_range* If *pos* > size().  
*std::length\_error* If new length exceeds `max_size()`.

Removes the characters in the range [*pos*, *pos* + *n1*) from this string. In place, *n2* copies of *c* are inserted. If *pos* is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1490 of file `basic_string.h`.

**5.84.2.109** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace ( iterator __i1, iterator __i2, initializer_list<_CharT> __l ) [inline, inherited]`

Replace range of characters with `initializer_list`.

### Parameters

*i1* Iterator referencing start of range to replace.  
*i2* Iterator referencing end of range to replace.  
*l* The `initializer_list` of characters to insert.

### Returns

Reference to this string.

### Exceptions

*std::length\_error* If new length exceeds `max_size()`.

Removes the characters in the range [*i1*, *i2*). In place, characters in the range [*k1*, *k2*) are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1659 of file `basic_string.h`.

**5.84.2.110** `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace ( size_type __pos, size_type __n1, const _CharT * __s, size_type __n2 ) [inherited]`

Replace characters with value of a C substring.

#### Parameters

*pos* Index of first character to replace.  
*n1* Number of characters to be replaced.  
*s* C string to insert.  
*n2* Number of characters from *s* to use.

#### Returns

Reference to this string.

#### Exceptions

*std::out\_of\_range* If *pos1* > size().  
*std::length\_error* If new length exceeds `max_size()`.

Removes the characters in the range [*pos*, *pos* + *n1*) from this string. In place, the first *n2* characters of *s* are inserted, or all of *s* if *n2* is too large. If *pos* is beyond end of string, *out\_of\_range* is thrown. If the length of result exceeds `max_size()`, *length\_error* is thrown. The value of the string doesn't change if an error is thrown.

**5.84.2.111** `void std::basic_string<_CharT, _Traits, _Allocator>::reserve ( size_type __res_arg = 0 ) [inherited]`

Attempt to preallocate enough memory for specified number of characters.

#### Parameters

*res\_arg* Number of characters required.

#### Exceptions

*std::length\_error* If *res\_arg* exceeds `max_size()`.

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than `max_size()`, *length\_error* is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

**5.84.2.112** `void std::basic_string<_CharT, _Traits, _Allocator>::resize (`  
`size_type __n, _CharT __c ) [inherited]`

Resizes the string to the specified number of characters.

#### Parameters

- n* Number of characters the string should contain.
- c* Character to fill any new elements.

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to *c*.

**5.84.2.113** `void std::basic_string<_CharT, _Traits, _Allocator>::resize (`  
`size_type __n ) [inline, inherited]`

Resizes the string to the specified number of characters.

#### Parameters

- n* Number of characters the string should contain.

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as `char`, this means setting them to 0.

Definition at line 748 of file `basic_string.h`.

**5.84.2.114** `size_type std::basic_string<_CharT, _Traits, _Allocator>::rfind`  
`( const basic_string<_CharT, _Traits, _Allocator> & __str,`  
`size_type __pos = npos ) const [inline, inherited]`

Find last position of a string.



**Parameters**

*str* String to locate.

*pos* Index of character to search back from (default end).

**Returns**

Index of start of last occurrence.

Starting from *pos*, searches backward for value of *str* within this string. If found, returns the index where it begins. If not found, returns *npos*.

Definition at line 1855 of file `basic_string.h`.

**5.84.2.115** `size_type std::basic_string<_CharT, _Traits, _Allocator>::rfind (`  
`_CharT __c, size_type __pos = npos ) const [inherited]`

Find last position of a character.

**Parameters**

*c* Character to locate.

*pos* Index of character to search back from (default end).

**Returns**

Index of last occurrence.

Starting from *pos*, searches backward for *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

**5.84.2.116** `size_type std::basic_string<_CharT, _Traits, _Allocator>::rfind (`  
`const _CharT* __s, size_type __pos = npos ) const [inline,`  
`inherited]`

Find last position of a C string.

**Parameters**

*s* C string to locate.

*pos* Index of character to start search at (default end).

**Returns**

Index of start of last occurrence.

Starting from *pos*, searches backward for the value of *s* within this string. If found, returns the index where it begins. If not found, returns *npos*.

Definition at line 1883 of file `basic_string.h`.

**5.84.2.117** `size_type std::basic_string<_CharT, _Traits, _Allocator>::rfind`  
( `const _CharT*` *s*, `size_type` *pos*, `size_type` *n* ) `const`  
[*inherited*]

Find last position of a C substring.

**Parameters**

*s* C string to locate.

*pos* Index of character to search back from.

*n* Number of characters from *s* to search for.

**Returns**

Index of start of last occurrence.

Starting from *pos*, searches backward for the first *n* characters in *s* within this string. If found, returns the index where it begins. If not found, returns *npos*.

**5.84.2.118** `void std::basic_string<_CharT, _Traits, _Allocator>::shrink_to_fit`  
( ) [*inline, inherited*]

A non-binding request to reduce `capacity()` to `size()`.

Definition at line 754 of file `basic_string.h`.

**5.84.2.119** `size_type std::basic_string<_CharT, _Traits, _Allocator>::size` ( )  
`const` [*inline, inherited*]

Returns the number of characters in the string, not including any /// null-termination.

Definition at line 710 of file `basic_string.h`.

**5.84.2.120** `basic_string` `std::basic_string<_CharT, _Traits, _Allocator>::substr ( size_type __pos = 0, size_type __n = npos ) const`  
[inline, inherited]

Get a substring.

#### Parameters

*pos* Index of first character (default 0).  
*n* Number of characters in substring (default remainder).

#### Returns

The new string.

#### Exceptions

[\*std::out\\_of\\_range\*](#) If *pos* > *size()*.

Construct and return a new string using the *n* characters starting at *pos*. If the string is too short, use the remainder of the characters. If *pos* is beyond the end of the string, *out\_of\_range* is thrown.

Definition at line 2155 of file `basic_string.h`.

**5.84.2.121** `void` `std::basic_string<_CharT, _Traits, _Allocator>::swap`  
( `basic_string<_CharT, _Traits, _Allocator> & __s` )  
[inherited]

Swap contents with another string.

#### Parameters

*s* String to swap with.

Exchanges the contents of this string with that of *s* in constant time.

### 5.84.3 Member Data Documentation

**5.84.3.1** `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators` [inherited]

## 5.85 `__gnu_parallel::__accumulate_binop_reduct<_BinOp>` Struct Template Reference 1161

---

The list of constant iterators that reference this container.

Definition at line 185 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if()`, and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_from_if()`.

### 5.84.3.2 `_Safe_iterator_base*` `__gnu_debug::_Safe_sequence_base::_M_iterators` [inherited]

The list of mutable iterators that reference this container.

Definition at line 182 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if()`, and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_from_if()`.

### 5.84.3.3 `unsigned int` `__gnu_debug::_Safe_sequence_base::_M_version` [mutable, inherited]

The container version number. This number may never be 0.

Definition at line 188 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`, and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_from_if()`.

### 5.84.3.4 `const size_type` `std::basic_string<_CharT, _Traits, _Allocator>::npos` [static, inherited]

Value returned by various member functions when they fail.

Definition at line 280 of file `basic_string.h`.

The documentation for this class was generated from the following file:

- [debug/string](#)

## 5.85 `__gnu_parallel::__accumulate_binop_reduct<_BinOp>` Struct Template Reference

General reduction, using a binary operator.

## 5.86 `__gnu_parallel::__accumulate_selector<_It>` Struct Template Reference

### Public Member Functions

- `__accumulate_binop_reduct` (`_BinOp` & `__b`)
- `template<typename _Result, typename _Addend>`  
`_Result operator()` (`const _Result` & `__x`, `const _Addend` & `__y`)

### Public Attributes

- `_BinOp` & `__binop`

#### 5.85.1 Detailed Description

`template<typename _BinOp> struct __gnu_parallel::__accumulate_binop_reduct<_BinOp>`

General reduction, using a binary operator.

Definition at line 335 of file `for_each_selectors.h`.

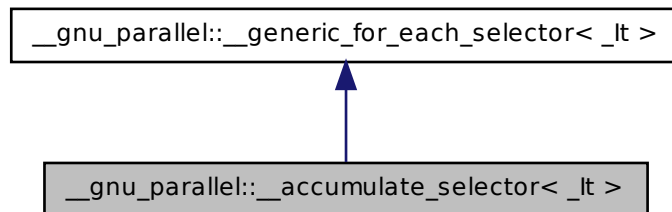
The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.86 `__gnu_parallel::__accumulate_selector<_It>` Struct Template Reference

[std::accumulate\(\)](#) selector.

Inheritance diagram for `__gnu_parallel::__accumulate_selector<_It>`:



## 5.86 `__gnu_parallel::__accumulate_selector<_It>` Struct Template Reference

### Public Member Functions

- `template<typename _Op>  
std::iterator_traits<_It>::value_type operator() (_Op __o, _It __i)`

### Public Attributes

- `_It __M_finish_iterator`

#### 5.86.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__accumulate_selector<_It>`

`std::accumulate()` selector.

Definition at line 208 of file `for_each_selectors.h`.

#### 5.86.2 Member Function Documentation

**5.86.2.1** `template<typename _It> template<typename _Op> std::iterator_traits<_It>::value_type __gnu_parallel::__accumulate_selector<_It>::operator() ( _Op __o, _It __i ) [inline]`

Functor execution.

##### Parameters

- `__o` Operator (unused).
- `__i` iterator referencing object.

##### Returns

The current value.

Definition at line 216 of file `for_each_selectors.h`.

#### 5.86.3 Member Data Documentation

**5.86.3.1** `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::__M_finish_iterator [inherited]`

## 5.87 `__gnu_parallel::__adjacent_difference_selector<_It>` Struct Template Reference 1164

---

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

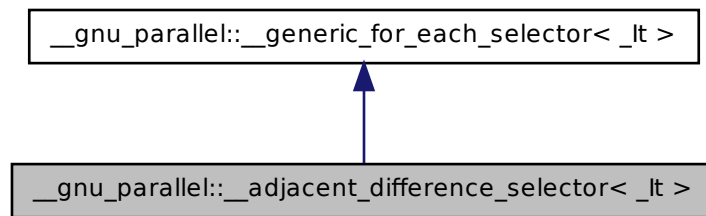
The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.87 `__gnu_parallel::__adjacent_difference_selector<_It>` Struct Template Reference

Selector that returns the difference between two adjacent `__elements`.

Inheritance diagram for `__gnu_parallel::__adjacent_difference_selector<_It>`:



### Public Member Functions

- `template<typename _Op>`  
`bool operator() (_Op &__o, _It __i)`

### Public Attributes

- `_It _M_finish_iterator`

### 5.87.1 Detailed Description

**template<typename `_It`> struct `__gnu_parallel::__adjacent_difference_selector<_It>`**

Selector that returns the difference between two adjacent `__elements`.

Definition at line 269 of file `for_each_selectors.h`.

### 5.87.2 Member Data Documentation

**5.87.2.1 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::M_finish_iterator [inherited]`**

`_Iterator` on last element processed; needed for some algorithms (e.g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

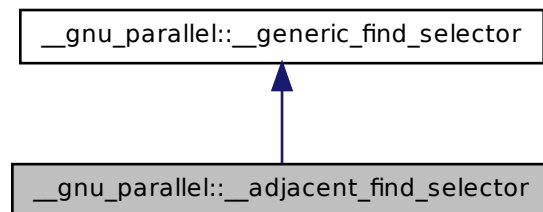
The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.88 `__gnu_parallel::__adjacent_find_selector` Struct Reference

Test predicate on two adjacent elements.

Inheritance diagram for `__gnu_parallel::__adjacent_find_selector`:





## Public Member Functions

- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`  
`std::pair< _RAIter1, _RAIter2 > \_M\_sequential\_algorithm (_RAIter1 __-`  
`begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`  
`bool operator\(\) (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`

### 5.88.1 Detailed Description

Test predicate on two adjacent elements.

Definition at line 80 of file `find_selectors.h`.

### 5.88.2 Member Function Documentation

**5.88.2.1** `template<typename _RAIter1, typename _RAIter2, typename _Pred > std::pair<_RAIter1, _RAIter2>`  
`__gnu_parallel::__adjacent_find_selector::_M_sequential_algorithm (`  
`__RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred`  
`__pred ) [inline]`

Corresponding sequential algorithm on a sequence.

#### Parameters

- `__begin1` Begin iterator of first sequence.
- `__end1` End iterator of first sequence.
- `__begin2` Begin iterator of second sequence.
- `__pred` Find predicate.

Definition at line 105 of file `find_selectors.h`.

References `std::make_pair()`.

**5.88.2.2** `template<typename _RAIter1, typename _RAIter2, typename _Pred > bool __gnu_parallel::__adjacent_find_selector::operator() (`  
`__RAIter1 __i1, _RAIter2 __i2, _Pred __pred ) [inline]`

Test on one position.

## 5.89 `__gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >` Class Template Reference 1167

### Parameters

- `__i1` Iterator on first sequence.
- `__i2` Iterator on second sequence (unused).
- `__pred` Find predicate.

Definition at line 90 of file `find_selectors.h`.

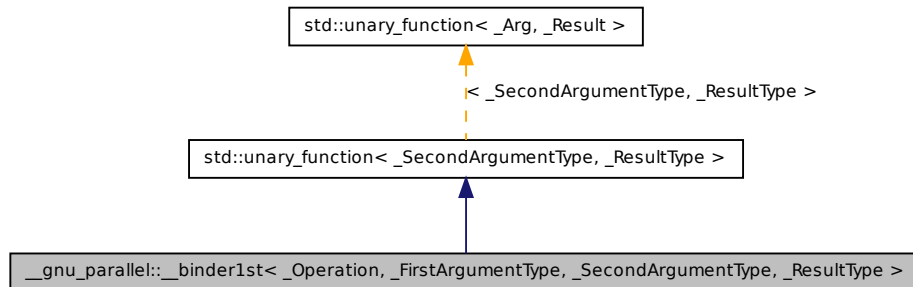
The documentation for this struct was generated from the following file:

- [find\\_selectors.h](#)

## 5.89 `__gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >` Class Template Reference

Similar to [std::binder1st](#), but giving the argument types explicitly.

Inheritance diagram for `__gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >`:



### Public Types

- typedef `_SecondArgumentType` [argument\\_type](#)
- typedef `_ResultType` [result\\_type](#)

### Public Member Functions

- `__binder1st` (const `_Operation` &\_\_x, const `_FirstArgumentType` &\_\_y)

## 5.89 `__gnu_parallel::__binder1st<_Operation, _FirstArgumentType, _SecondArgumentType, _ResultType>` Class Template Reference 1168

---

- `_ResultType operator() (_SecondArgumentType &__x) const`
- `_ResultType operator() (const _SecondArgumentType &__x)`

### Protected Attributes

- `_Operation _M_op`
- `_FirstArgumentType _M_value`

### 5.89.1 Detailed Description

`template<typename _Operation, typename _FirstArgumentType, typename _SecondArgumentType, typename _ResultType> class __gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >`

Similar to [std::binder1st](#), but giving the argument types explicitly.

Definition at line 192 of file `parallel/base.h`.

### 5.89.2 Member Typedef Documentation

**5.89.2.1** `typedef _SecondArgumentType std::unary_function< _SecondArgumentType , _ResultType >::argument_type [inherited]`

`argument_type` is the type of the argument

Definition at line 105 of file `stl_function.h`.

**5.89.2.2** `typedef _ResultType std::unary_function< _SecondArgumentType , _ResultType >::result_type [inherited]`

`result_type` is the return type

Definition at line 108 of file `stl_function.h`.

The documentation for this class was generated from the following file:

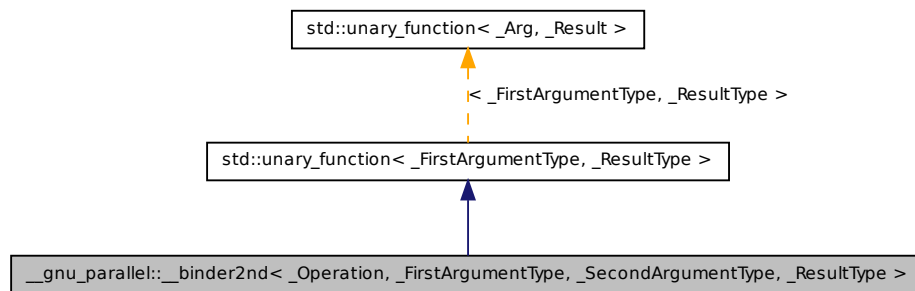
- [parallel/base.h](#)

## 5.90 `__gnu_parallel::__binder2nd<_Operation, _FirstArgumentType, _SecondArgumentType, _ResultType>` Class Template Reference 1169

### 5.90 `__gnu_parallel::__binder2nd<_Operation, _FirstArgumentType, _SecondArgumentType, _ResultType>` Class Template Reference

Similar to [std::binder2nd](#), but giving the argument types explicitly.

Inheritance diagram for `__gnu_parallel::__binder2nd<_Operation, _FirstArgumentType, _SecondArgumentType, _ResultType>`:



#### Public Types

- typedef `_FirstArgumentType` [argument\\_type](#)
- typedef `_ResultType` [result\\_type](#)

#### Public Member Functions

- `__binder2nd` (`const _Operation &__x, const _SecondArgumentType &__y`)
- `_ResultType operator()` (`_FirstArgumentType &__x`)
- `_ResultType operator()` (`const _FirstArgumentType &__x`) `const`

#### Protected Attributes

- `_Operation` `_M_op`
- `_SecondArgumentType` `_M_value`

### 5.90.1 Detailed Description

```
template<typename _Operation, typename _FirstArgumentType, typename
_SecondArgumentType, typename _ResultType> class __gnu_parallel::__-
binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _-
ResultType >
```

Similar to [std::binder2nd](#), but giving the argument types explicitly.

Definition at line 220 of file `parallel/base.h`.

### 5.90.2 Member Typedef Documentation

**5.90.2.1** `typedef _FirstArgumentType std::unary_function<_FirstArgumentType , _ResultType >::argument_type [inherited]`

`argument_type` is the type of the argument

Definition at line 105 of file `stl_function.h`.

**5.90.2.2** `typedef _ResultType std::unary_function<_FirstArgumentType , _ResultType >::result_type [inherited]`

`result_type` is the return type

Definition at line 108 of file `stl_function.h`.

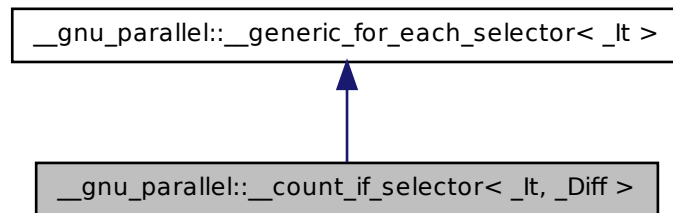
The documentation for this class was generated from the following file:

- [parallel/base.h](#)

## 5.91 `__gnu_parallel::__count_if_selector<_It, _Diff>` Struct Template Reference

`std::count_if()` selector.

Inheritance diagram for `__gnu_parallel::__count_if_selector<_It, _Diff>`:



## Public Member Functions

- `template<typename _Op> _Diff operator() (_Op &__o, _It __i)`

## Public Attributes

- `_It _M_finish_iterator`

### 5.91.1 Detailed Description

`template<typename _It, typename _Diff> struct __gnu_parallel::__count_if_selector<_It, _Diff>`

`std::count_if()` selector.

Definition at line 194 of file `for_each_selectors.h`.

### 5.91.2 Member Function Documentation

**5.91.2.1** `template<typename _It, typename _Diff> template<typename _Op> > _Diff __gnu_parallel::__count_if_selector<_It, _Diff>::operator() ( _Op & __o, _It __i ) [inline]`

Functor execution.

## 5.92 `__gnu_parallel::__count_selector<_It, _Diff>` Struct Template Reference 1172

### Parameters

- `__o` Operator.
- `__i` iterator referencing object.

### Returns

1 if count, 0 if does not count.

Definition at line 202 of file `for_each_selectors.h`.

### 5.91.3 Member Data Documentation

#### 5.91.3.1 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::__M_finish_iterator` [`inherited`]

`_Iterator` on last element processed; needed for some algorithms (e.g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

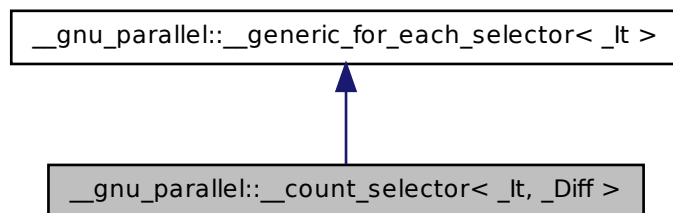
- [for\\_each\\_selectors.h](#)

## 5.92 `__gnu_parallel::__count_selector<_It, _Diff>` Struct Template Reference

`std::count()` selector.

## 5.92 `__gnu_parallel::__count_selector<_It, _Diff>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__count_selector<_It, _Diff>`:



### Public Member Functions

- `template<typename _ValueType > _Diff operator() (_ValueType &__v, _It __i)`

### Public Attributes

- `_It _M_finish_iterator`

#### 5.92.1 Detailed Description

`template<typename _It, typename _Diff> struct __gnu_parallel::__count_selector<_It, _Diff>`

`std::count()` selector.

Definition at line 180 of file `for_each_selectors.h`.

#### 5.92.2 Member Function Documentation

**5.92.2.1** `template<typename _It, typename _Diff> template<typename _ValueType > _Diff __gnu_parallel::__count_selector<_It, _Diff>::operator() ( _ValueType & __v, _It __i ) [inline]`

Functor execution.



## 5.93 `__gnu_parallel::__fill_selector<_It>` Struct Template Reference 1174

---

### Parameters

- `__v` Current value.
- `__i` iterator referencing object.

### Returns

- 1 if count, 0 if does not count.

Definition at line 188 of file `for_each_selectors.h`.

### 5.92.3 Member Data Documentation

#### 5.92.3.1 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::__M_finish_iterator` [`inherited`]

`_Iterator` on last element processed; needed for some algorithms (e.g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

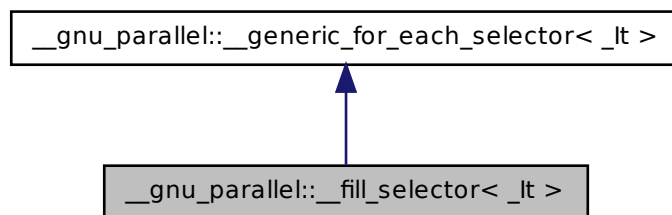
The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.93 `__gnu_parallel::__fill_selector<_It>` Struct Template Reference

`std::fill()` selector.

Inheritance diagram for `__gnu_parallel::__fill_selector<_It>`:



### Public Member Functions

- `template<typename _ValueType > bool operator() (_ValueType &__v, _It __i)`

### Public Attributes

- `_It _M_finish_iterator`

#### 5.93.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__fill_selector<_It>`

`std::fill()` selector.

Definition at line 84 of file `for_each_selectors.h`.

#### 5.93.2 Member Function Documentation

**5.93.2.1** `template<typename _It > template<typename _ValueType > bool __gnu_parallel::__fill_selector<_It>::operator() ( _ValueType & __v, _It __i ) [inline]`

Functor execution.

## 5.94 `__gnu_parallel::__find_first_of_selector<_FIterator>` Struct Template Reference 1176

---

### Parameters

- `__v` Current value.
- `__i` iterator referencing object.

Definition at line 91 of file `for_each_selectors.h`.

### 5.93.3 Member Data Documentation

#### 5.93.3.1 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::__M_finish_iterator` [`inherited`]

`_Iterator` on last element processed; needed for some algorithms (e.g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

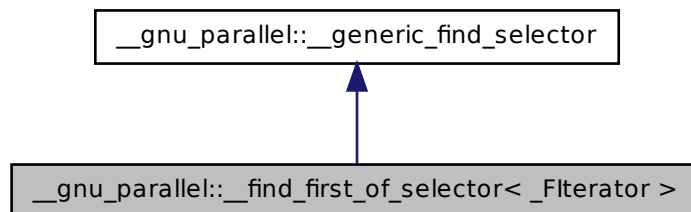
The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.94 `__gnu_parallel::__find_first_of_selector<_FIterator>` Struct Template Reference

Test predicate on several elements.

Inheritance diagram for `__gnu_parallel::__find_first_of_selector<_FIterator>`:



## Public Member Functions

- `__find_first_of_selector` (`_FIterator __begin`, `_FIterator __end`)
- `template<typename _RAIter1, typename _RAIter2, typename _Pred>`  
`std::pair<_RAIter1, _RAIter2> __M_sequential_algorithm` (`_RAIter1 __begin1`, `_RAIter1 __end1`, `_RAIter2 __begin2`, `_Pred __pred`)
- `template<typename _RAIter1, typename _RAIter2, typename _Pred>`  
`bool operator()` (`_RAIter1 __i1`, `_RAIter2 __i2`, `_Pred __pred`)

## Public Attributes

- `_FIterator __M_begin`
- `_FIterator __M_end`

### 5.94.1 Detailed Description

`template<typename _FIterator> struct __gnu_parallel::__find_first_of_selector<_FIterator>`

Test predicate on several elements.

Definition at line 153 of file `find_selectors.h`.

### 5.94.2 Member Function Documentation

**5.94.2.1** `template<typename _FIterator> template<typename _RAIter1, typename _RAIter2, typename _Pred> std::pair<_RAIter1, _RAIter2> __gnu_parallel::__find_first_of_selector<_FIterator>::__M_sequential_algorithm` (`_RAIter1 __begin1`, `_RAIter1 __end1`, `_RAIter2 __begin2`, `_Pred __pred`) [`inline`]

Corresponding sequential algorithm on a sequence.

#### Parameters

- `__begin1` Begin iterator of first sequence.
- `__end1` End iterator of first sequence.
- `__begin2` Begin iterator of second sequence.
- `__pred` Find predicate.

Definition at line 186 of file `find_selectors.h`.

References `std::make_pair()`.

**5.94.2.2** `template<typename _FIterator > template<typename  
_RAIter1 , typename _RAIter2 , typename _Pred > bool  
__gnu_parallel::__find_first_of_selector< _FIterator >::operator() (  
_RAIter1 __i1, _RAIter2 __i2, _Pred __pred ) [inline]`

Test on one position.

#### Parameters

- `__i1` Iterator on first sequence.
- `__i2` Iterator on second sequence (unused).
- `__pred` Find predicate.

Definition at line 169 of file `find_selectors.h`.

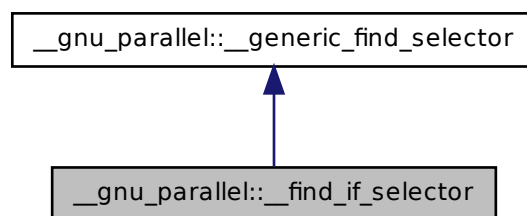
The documentation for this struct was generated from the following file:

- [find\\_selectors.h](#)

## 5.95 \_\_gnu\_parallel::\_\_find\_if\_selector Struct Reference

Test predicate on a single element, used for `std::find()` and `std::find_if()`.

Inheritance diagram for `__gnu_parallel::__find_if_selector`:



#### Public Member Functions

- `template<typename _RAIter1 , typename _RAIter2 , typename _Pred >`

```
std::pair<_RAIter1, _RAIter2 > _M_sequential_algorithm (_RAIter1 __-
begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)
```

- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`  
`bool operator() (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`

### 5.95.1 Detailed Description

Test predicate on a single element, used for `std::find()` and `std::find_if()`.

Definition at line 50 of file `find_selectors.h`.

### 5.95.2 Member Function Documentation

**5.95.2.1** `template<typename _RAIter1, typename _RAIter2, typename _Pred > std::pair<_RAIter1, _RAIter2>`  
`__gnu_parallel::__find_if_selector::_M_sequential_algorithm (`  
`_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred`  
`__pred ) [inline]`

Corresponding sequential algorithm on a sequence.

#### Parameters

- `__begin1` Begin iterator of first sequence.
- `__end1` End iterator of first sequence.
- `__begin2` Begin iterator of second sequence.
- `__pred` Find predicate.

Definition at line 72 of file `find_selectors.h`.

References `std::make_pair()`.

**5.95.2.2** `template<typename _RAIter1, typename _RAIter2, typename _Pred`  
`> bool __gnu_parallel::__find_if_selector::operator() ( _RAIter1`  
`__i1, _RAIter2 __i2, _Pred __pred ) [inline]`

Test on one position.

#### Parameters

- `__i1` Iterator on first sequence.

## 5.96 `__gnu_parallel::__for_each_selector<_It>` Struct Template Reference 180

`__i2` Iterator on second sequence (unused).

`__pred` Find predicate.

Definition at line 60 of file `find_selectors.h`.

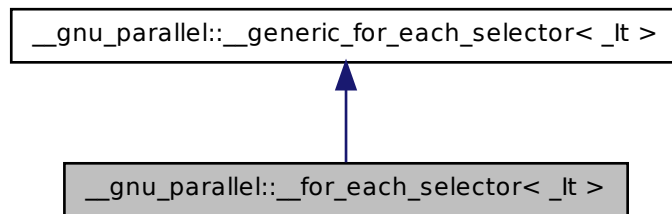
The documentation for this struct was generated from the following file:

- [find\\_selectors.h](#)

## 5.96 `__gnu_parallel::__for_each_selector<_It>` Struct Template Reference

`std::for_each()` selector.

Inheritance diagram for `__gnu_parallel::__for_each_selector<_It>`:



### Public Member Functions

- `template<typename _Op>`  
`bool operator() (_Op &__o, _It __i)`

### Public Attributes

- `_It __M_finish_iterator`

## 5.97 `__gnu_parallel::__generate_selector<_It>` Struct Template Reference 181

### 5.96.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__for_each_selector<_It>`

`std::for_each()` selector.

Definition at line 52 of file `for_each_selectors.h`.

### 5.96.2 Member Function Documentation

**5.96.2.1** `template<typename _It> template<typename _Op> bool  
__gnu_parallel::__for_each_selector<_It>::operator()( _Op & __o,  
_It __i ) [inline]`

Functor execution.

#### Parameters

`__o` Operator.

`__i` iterator referencing object.

Definition at line 59 of file `for_each_selectors.h`.

### 5.96.3 Member Data Documentation

**5.96.3.1** `template<typename _It> _It __gnu_parallel::__-  
generic_for_each_selector<_It>::__M_finish_iterator  
[inherited]`

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

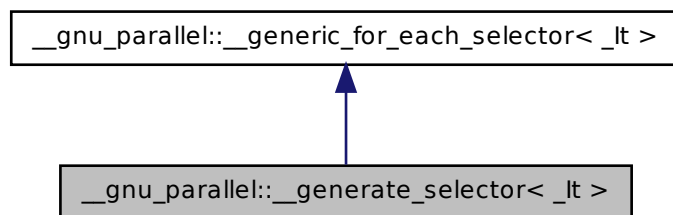
## 5.97 `__gnu_parallel::__generate_selector<_It>` Struct Template Reference

`std::generate()` selector.



## 5.97 `__gnu_parallel::__generate_selector<_It>` Struct Template Reference 182

Inheritance diagram for `__gnu_parallel::__generate_selector<_It>`:



### Public Member Functions

- `template<typename _Op>`  
`bool operator() (_Op &__o, _It __i)`

### Public Attributes

- `_It _M_finish_iterator`

#### 5.97.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__generate_selector<_It>`

`std::generate()` selector.

Definition at line 68 of file `for_each_selectors.h`.

#### 5.97.2 Member Function Documentation

**5.97.2.1** `template<typename _It> template<typename _Op> bool`  
`__gnu_parallel::__generate_selector<_It>::operator() ( _Op & __o,`  
`_It __i ) [inline]`

Functor execution.

**Parameters**

- `__o` Operator.
- `__i` iterator referencing object.

Definition at line 75 of file `for_each_selectors.h`.

**5.97.3 Member Data Documentation**
**5.97.3.1** `template<typename _It > _It __gnu_parallel::__generic_for_each_selector< _It >::__M_finish_iterator [inherited]`

`_Iterator` on last element processed; needed for some algorithms (e.g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

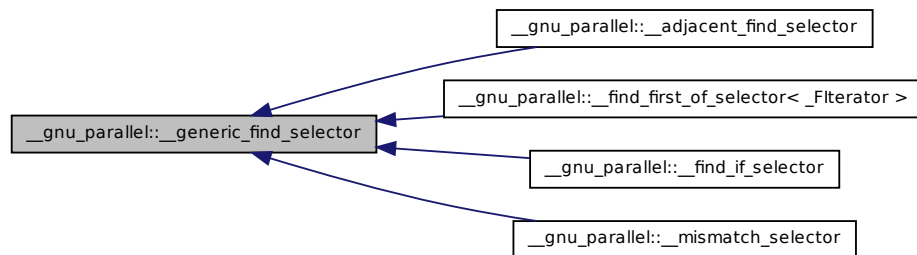
The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

**5.98 \_\_gnu\_parallel::\_\_generic\_find\_selector Struct Reference**

Base class of all [\\_\\_gnu\\_parallel::\\_\\_find\\_template](#) selectors.

Inheritance diagram for `__gnu_parallel::__generic_find_selector`:



## 5.99 `__gnu_parallel::__generic_for_each_selector<_It>` Struct Template Reference 1184

---

### 5.98.1 Detailed Description

Base class of all `__gnu_parallel::__find_template` selectors.

Definition at line 43 of file `find_selectors.h`.

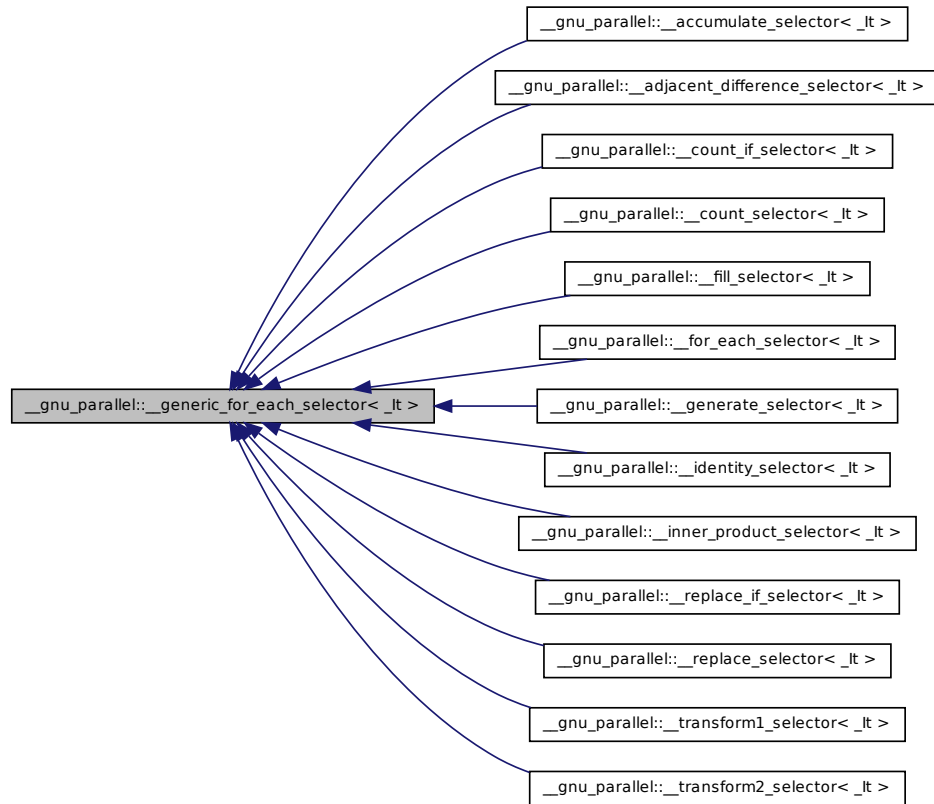
The documentation for this struct was generated from the following file:

- [find\\_selectors.h](#)

## 5.99 `__gnu_parallel::__generic_for_each_selector<_It>` Struct Template Reference

Generic `__selector` for embarrassingly parallel functions.

Inheritance diagram for `__gnu_parallel::__generic_for_each_selector<_It>`:



## Public Attributes

- `_It` [\\_M\\_finish\\_iterator](#)

### 5.99.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__generic_for_each_selector<_It>`

Generic `__selector` for embarrassingly parallel functions.

Definition at line 42 of file `for_each_selectors.h`.

## 5.100 `__gnu_parallel::__identity_selector<_It>` Struct Template Reference

### 5.99.2 Member Data Documentation

#### 5.99.2.1 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::__M_finish_iterator`

`_Iterator` on last element processed; needed for some algorithms (e.g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

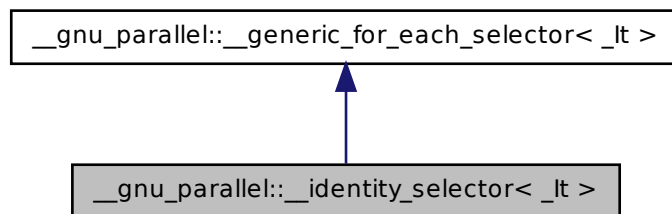
The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.100 `__gnu_parallel::__identity_selector<_It>` Struct Template Reference

Selector that just returns the passed iterator.

Inheritance diagram for `__gnu_parallel::__identity_selector<_It>`:



### Public Member Functions

- `template<typename _Op> _It operator() (_Op __o, _It __i)`

### Public Attributes

- `_It M\_finish\_iterator`

## 5.100 `__gnu_parallel::__identity_selector<_It>` Struct Template Reference

### 5.100.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__identity_selector<_It>`

Selector that just returns the passed iterator.

Definition at line 253 of file `for_each_selectors.h`.

### 5.100.2 Member Function Documentation

**5.100.2.1** `template<typename _It> template<typename _Op> _It  
__gnu_parallel::__identity_selector<_It>::operator()( _Op __o,  
_It __i ) [inline]`

Functor execution.

#### Parameters

`__o` Operator (unused).

`__i` iterator referencing object.

#### Returns

Passed iterator.

Definition at line 261 of file `for_each_selectors.h`.

### 5.100.3 Member Data Documentation

**5.100.3.1** `template<typename _It> _It __gnu_parallel::__-  
generic_for_each_selector<_It>::_M_finish_iterator  
[inherited]`

`_Iterator` on last element processed; needed for some algorithms (e.g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

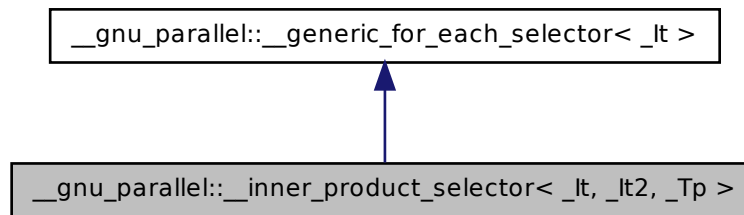
The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.101 `__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>` Struct Template Reference

[std::inner\\_product\(\)](#) selector.

Inheritance diagram for `__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>`:



### Public Member Functions

- [\\_\\_inner\\_product\\_selector](#) (`_It __b1, _It2 __b2`)
- `template<typename _Op>`  
`_Tp operator() (_Op __mult, _It __current)`

### Public Attributes

- `_It __begin1_iterator`
- `_It2 __begin2_iterator`
- `_It __M_finish_iterator`

#### 5.101.1 Detailed Description

`template<typename _It, typename _It2, typename _Tp> struct __gnu_parallel::__inner_product_selector<_It, _It2, _Tp>`

[std::inner\\_product\(\)](#) selector.

Definition at line 222 of file `for_each_selectors.h`.

### 5.101.2 Constructor & Destructor Documentation

5.101.2.1 `template<typename _It, typename _It2, typename _Tp>  
__gnu_parallel::__inner_product_selector<_It, _It2, _Tp  
>::__inner_product_selector( _It __b1, _It2 __b2 ) [inline,  
explicit]`

Constructor.

#### Parameters

- b1* Begin iterator of first sequence.
- b2* Begin iterator of second sequence.

Definition at line 234 of file `for_each_selectors.h`.

### 5.101.3 Member Function Documentation

5.101.3.1 `template<typename _It, typename _It2, typename _Tp>  
template<typename _Op> _Tp __gnu_parallel::__inner_product_  
selector<_It, _It2, _Tp>::operator()( _Op __mult, _It __current )  
[inline]`

Functor execution.

#### Parameters

- \_\_mult* Multiplication functor.
- \_\_current* iterator referencing object.

#### Returns

Inner product elemental `__result`.

Definition at line 243 of file `for_each_selectors.h`.

References `__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>::__begin1_`-iterator, and `__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>::__begin2_`-iterator.



#### 5.101.4 Member Data Documentation

**5.101.4.1** `template<typename _It, typename _It2, typename _Tp>  
_It __gnu_parallel::__inner_product_selector<_It, _It2, _Tp>::__begin1_iterator`

Begin iterator of first sequence.

Definition at line 225 of file `for_each_selectors.h`.

Referenced by `__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>::operator()`.

**5.101.4.2** `template<typename _It, typename _It2, typename _Tp>  
_It2 __gnu_parallel::__inner_product_selector<_It, _It2, _Tp>::__begin2_iterator`

Begin iterator of second sequence.

Definition at line 228 of file `for_each_selectors.h`.

Referenced by `__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>::operator()`.

**5.101.4.3** `template<typename _It> _It __gnu_parallel::__-  
generic_for_each_selector<_It>::__M_finish_iterator  
[inherited]`

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.102 `__gnu_parallel::__max_element_reduct<_Compare, _It>` Struct Template Reference

Reduction for finding the maximum element, using a comparator.

#### Public Member Functions

- `__max_element_reduct` (`_Compare &__c`)
- `_It operator()` (`_It __x, _It __y`)

#### Public Attributes

- `_Compare & __comp`

##### 5.102.1 Detailed Description

`template<typename _Compare, typename _It> struct __gnu_parallel::__max_element_reduct<_Compare, _It>`

Reduction for finding the maximum element, using a comparator.

Definition at line 321 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

#### 5.103 `__gnu_parallel::__min_element_reduct<_Compare, _It> Struct` Template Reference

Reduction for finding the maximum element, using a comparator.

#### Public Member Functions

- `__min_element_reduct` (`_Compare &__c`)
- `_It operator()` (`_It __x, _It __y`)

#### Public Attributes

- `_Compare & __comp`

##### 5.103.1 Detailed Description

`template<typename _Compare, typename _It> struct __gnu_parallel::__min_element_reduct<_Compare, _It>`

Reduction for finding the maximum element, using a comparator.

Definition at line 307 of file `for_each_selectors.h`.

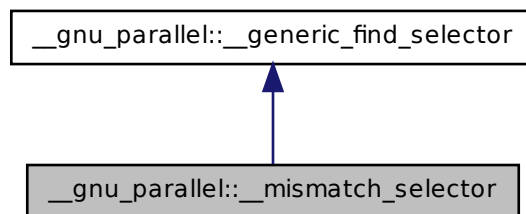
The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.104 `__gnu_parallel::__mismatch_selector` Struct Reference

Test inverted predicate on a single element.

Inheritance diagram for `__gnu_parallel::__mismatch_selector`:



### Public Member Functions

- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`  
`std::pair< _RAIter1, _RAIter2 > \_M\_sequential\_algorithm (_RAIter1 __-`  
`begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`  
`bool operator\(\) (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`

#### 5.104.1 Detailed Description

Test inverted predicate on a single element.

Definition at line 119 of file `find_selectors.h`.

#### 5.104.2 Member Function Documentation

**5.104.2.1** `template<typename _RAIter1, typename _RAIter2, typename _Pred > std::pair<_RAIter1, _RAIter2> __gnu_parallel::__mismatch_selector::__M_sequential_algorithm ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred ) [inline]`

Corresponding sequential algorithm on a sequence.

##### Parameters

`__begin1` Begin iterator of first sequence.  
`__end1` End iterator of first sequence.  
`__begin2` Begin iterator of second sequence.  
`__pred` Find predicate.

Definition at line 143 of file `find_selectors.h`.

**5.104.2.2** `template<typename _RAIter1, typename _RAIter2, typename _Pred > bool __gnu_parallel::__mismatch_selector::operator() ( _RAIter1 __i1, _RAIter2 __i2, _Pred __pred ) [inline]`

Test on one position.

##### Parameters

`__i1` Iterator on first sequence.  
`__i2` Iterator on second sequence (unused).  
`__pred` Find predicate.

Definition at line 130 of file `find_selectors.h`.

The documentation for this struct was generated from the following file:

- [find\\_selectors.h](#)

**5.105** `__gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct` Template Reference

Switch for 3-way merging with `__sentinels` turned off.

## 5.106 `__gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >` Struct Template Reference 1194

### Public Member Functions

- `_RAIter3 operator() (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`

#### 5.105.1 Detailed Description

`template<bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare> struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`

Switch for 3-way merging with `__sentinels` turned off. Note that 3-way merging is always stable!

Definition at line 748 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

## 5.106 `__gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >` Struct Template Reference

Switch for 3-way merging with `__sentinels` turned on.

### Public Member Functions

- `_RAIter3 operator() (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`

#### 5.106.1 Detailed Description

`template<typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare> struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`

Switch for 3-way merging with `__sentinels` turned on. Note that 3-way merging is always stable!

Definition at line 768 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

**5.107 `__gnu_parallel::__multiway_merge_4_variant_sentinel_switch<__sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct` Template Reference** 1195

---

- [multiway\\_merge.h](#)

**5.107 `__gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct` Template Reference**

Switch for 4-way merging with `__sentinels` turned off.

**Public Member Functions**

- `_RAIter3 operator() (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`

**5.107.1 Detailed Description**

`template<bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare> struct __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`

Switch for 4-way merging with `__sentinels` turned off. Note that 4-way merging is always stable!

Definition at line 791 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

**5.108 `__gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct` Template Reference**

Switch for 4-way merging with `__sentinels` turned on.

**Public Member Functions**

- `_RAIter3 operator() (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`

**5.109** `__gnu_parallel::__multiway_merge_k_variant_sentinel_switch<__sentinels, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`  
Struct Template Reference 1196

---

#### 5.108.1 Detailed Description

`template<typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare> struct __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`

Switch for 4-way merging with `__sentinels` turned on. Note that 4-way merging is always stable!

Definition at line 811 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

**5.109** `__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >` Struct Template Reference

Switch for k-way merging with `__sentinels` turned on.

#### Public Member Functions

- `_RAIter3 operator() (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)`

#### 5.109.1 Detailed Description

`template<bool __sentinels, bool __stable, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare> struct __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`

Switch for k-way merging with `__sentinels` turned on.

Definition at line 833 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

**5.110** `__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct`  
Template Reference 1197

**5.110** `__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct` Template Reference

Switch for k-way merging with `__sentinels` turned off.

#### Public Member Functions

- `_RAIter3 operator() (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)`

##### 5.110.1 Detailed Description

`template<bool __stable, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare> struct __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`

Switch for k-way merging with `__sentinels` turned off.

Definition at line 868 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

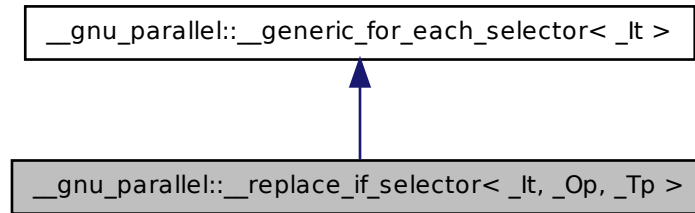
- [multiway\\_merge.h](#)

**5.111** `__gnu_parallel::__replace_if_selector< _It, _Op, _Tp >`  
Struct Template Reference

`std::replace()` selector.



Inheritance diagram for `__gnu_parallel::__replace_if_selector<_It, _Op, _Tp>`:



#### Public Member Functions

- `__replace_if_selector` (const `_Tp` & `__new_val`)
- bool `operator()` (`_Op` & `__o`, `_It` `__i`)

#### Public Attributes

- const `_Tp` & `__new_val`
- `_It` `__M_finish_iterator`

#### 5.111.1 Detailed Description

`template<typename _It, typename _Op, typename _Tp> struct __gnu_parallel::__replace_if_selector<_It, _Op, _Tp>`

`std::replace()` selector.

Definition at line 156 of file `for_each_selectors.h`.

#### 5.111.2 Constructor & Destructor Documentation

**5.111.2.1** `template<typename _It, typename _Op, typename _Tp>  
> __gnu_parallel::__replace_if_selector<_It, _Op, _Tp>:  
>::__replace_if_selector ( const _Tp & __new_val ) [inline,  
explicit]`

Constructor.

#### Parameters

*\_\_new\_val* Value to replace with.

Definition at line 164 of file `for_each_selectors.h`.

### 5.111.3 Member Function Documentation

**5.111.3.1** `template<typename _It, typename _Op, typename _Tp> bool  
__gnu_parallel::__replace_if_selector<_It, _Op, _Tp>::operator()(  
_Op & __o, _It __i ) [inline]`

Functor execution.

#### Parameters

*\_\_o* Operator.

*\_\_i* iterator referencing object.

Definition at line 170 of file `for_each_selectors.h`.

References `__gnu_parallel::__replace_if_selector<_It, _Op, _Tp>::__new_val`.

### 5.111.4 Member Data Documentation

**5.111.4.1** `template<typename _It, typename _Op, typename _Tp> const  
_Tp& __gnu_parallel::__replace_if_selector<_It, _Op, _Tp>::__new_val`

Value to replace with.

Definition at line 159 of file `for_each_selectors.h`.

Referenced by `__gnu_parallel::__replace_if_selector<_It, _Op, _Tp>::operator()()`.

**5.111.4.2** `template<typename _It> _It __gnu_parallel::__-  
generic_for_each_selector<_It>::__M_finish_iterator  
[inherited]`

### 5.112 `__gnu_parallel::__replace_selector<_It, _Tp>` Struct Template Reference

1200

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

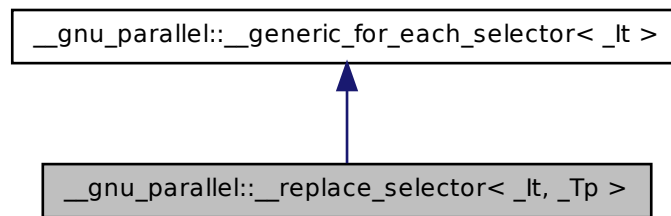
The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

### 5.112 `__gnu_parallel::__replace_selector<_It, _Tp>` Struct Template Reference

`std::replace()` selector.

Inheritance diagram for `__gnu_parallel::__replace_selector<_It, _Tp>`:



#### Public Member Functions

- [\\_\\_replace\\_selector](#) (const `_Tp` & [\\_\\_new\\_val](#))
- bool [operator\(\)](#) (`_Tp` & `__v`, `_It` `__i`)

#### Public Attributes

- const `_Tp` & [\\_\\_new\\_val](#)
- `_It` [\\_M\\_finish\\_iterator](#)

### 5.112.1 Detailed Description

**template<typename \_It, typename \_Tp> struct \_\_gnu\_parallel::\_\_replace\_selector<\_It, \_Tp>**

std::replace() selector.

Definition at line 132 of file for\_each\_selectors.h.

### 5.112.2 Constructor & Destructor Documentation

**5.112.2.1 template<typename \_It, typename \_Tp> \_\_gnu\_parallel::\_\_replace\_selector<\_It, \_Tp>::\_\_replace\_selector ( const \_Tp & \_\_new\_val ) [inline, explicit]**

Constructor.

#### Parameters

**\_\_new\_val** Value to replace with.

Definition at line 140 of file for\_each\_selectors.h.

### 5.112.3 Member Function Documentation

**5.112.3.1 template<typename \_It, typename \_Tp> bool \_\_gnu\_parallel::\_\_replace\_selector<\_It, \_Tp>::operator() ( \_Tp & \_\_v, \_It \_\_i ) [inline]**

Functor execution.

#### Parameters

**\_\_v** Current value.

**\_\_i** iterator referencing object.

Definition at line 146 of file for\_each\_selectors.h.

References `__gnu_parallel::__replace_selector<_It, _Tp>::__new_val`.

**5.112.4 Member Data Documentation**

**5.112.4.1 `template<typename _It, typename _Tp> const _Tp& __gnu_parallel::__replace_selector<_It, _Tp>::__new_val`**

Value to replace with.

Definition at line 135 of file `for_each_selectors.h`.

Referenced by `__gnu_parallel::__replace_selector<_It, _Tp>::operator()`.

**5.112.4.2 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::__M_finish_iterator [inherited]`**

`_Iterator` on last element processed; needed for some algorithms (e.g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

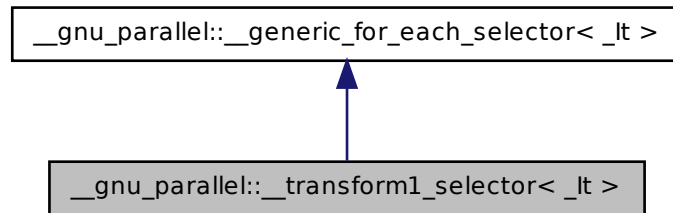
The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

**5.113 `__gnu_parallel::__transform1_selector<_It>` Struct Template Reference**

`std::transform()` `__selector`, one input sequence variant.

Inheritance diagram for `__gnu_parallel::__transform1_selector<_It>`:



## Public Member Functions

- `template<typename _Op>`  
`bool operator() (_Op &__o, _It __i)`

## Public Attributes

- `_It _M_finish_iterator`

### 5.113.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__transform1_selector<_It>`

`std::transform()` \_\_selector, one input sequence variant.

Definition at line 100 of file `for_each_selectors.h`.

### 5.113.2 Member Function Documentation

**5.113.2.1** `template<typename _It> template<typename _Op> bool`  
`__gnu_parallel::__transform1_selector<_It>::operator() ( _Op &`  
`__o, _It __i ) [inline]`

Functor execution.

#### Parameters

- `__o` Operator.
- `__i` iterator referencing object.

Definition at line 107 of file `for_each_selectors.h`.

#### 5.113.3 Member Data Documentation

##### 5.113.3.1 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::__M_finish_iterator` [inherited]

`_Iterator` on last element processed; needed for some algorithms (e.g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

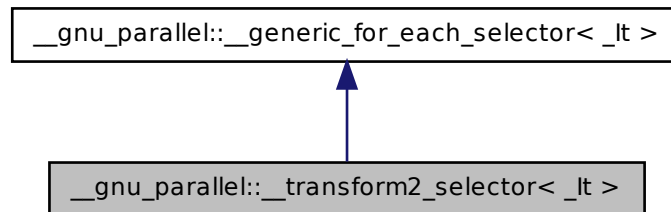
The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

#### 5.114 `__gnu_parallel::__transform2_selector<_It>` Struct Template Reference

`std::transform()` `__selector`, two input sequences variant.

Inheritance diagram for `__gnu_parallel::__transform2_selector<_It>`:



## Public Member Functions

- `template<typename _Op > bool operator() (_Op &__o, _It __i)`

## Public Attributes

- `_It _M_finish_iterator`

### 5.114.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__transform2_selector<_It>`

`std::transform()` \_\_selector, two input sequences variant.

Definition at line 116 of file `for_each_selectors.h`.

### 5.114.2 Member Function Documentation

**5.114.2.1** `template<typename _It> template<typename _Op > bool __gnu_parallel::__transform2_selector<_It>::operator() (_Op &__o, _It __i) [inline]`

Functor execution.

#### Parameters

- `__o` Operator.
- `__i` iterator referencing object.

Definition at line 123 of file `for_each_selectors.h`.

### 5.114.3 Member Data Documentation

**5.114.3.1** `template<typename _It > _It __gnu_parallel::__generic_for_each_selector<_It>::__M_finish_iterator [inherited]`

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).



### 5.115 `__gnu_parallel::__unary_negate< _Predicate, argument_type >` Class Template Reference 1206

---

Definition at line 47 of file `for_each_selectors.h`.

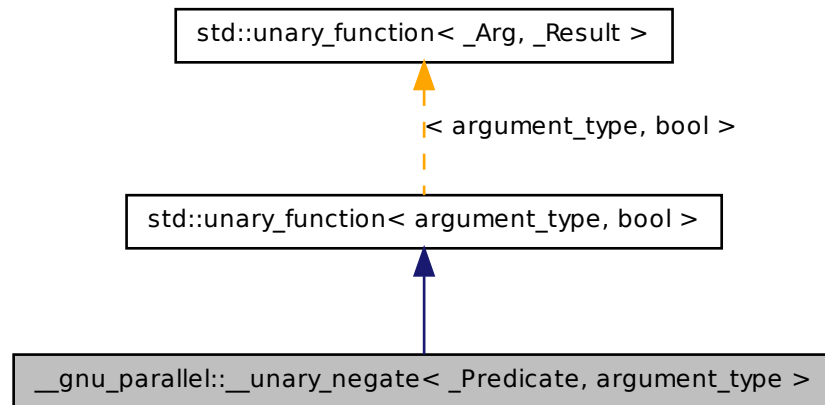
The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

### 5.115 `__gnu_parallel::__unary_negate< _Predicate, argument_type >` Class Template Reference

Similar to [std::unary\\_negate](#), but giving the argument types explicitly.

Inheritance diagram for `__gnu_parallel::__unary_negate< _Predicate, argument_type >`:



#### Public Types

- typedef [argument\\_type](#) `argument_type`
- typedef bool [result\\_type](#)

#### Public Member Functions

- `__unary_negate` (`const _Predicate &__x`)
- bool `operator()` (`const argument\_type &__x`)

#### Protected Attributes

- `_Predicate _M_pred`

#### 5.115.1 Detailed Description

`template<typename _Predicate, typename argument_type> class __gnu_parallel::_unary_negate<_Predicate, argument_type>`

Similar to [std::unary\\_negate](#), but giving the argument types explicitly.

Definition at line 173 of file `parallel/base.h`.

#### 5.115.2 Member Typedef Documentation

**5.115.2.1** `typedef argument_type std::unary_function< argument_type , bool >::argument_type [inherited]`

`argument_type` is the type of the argument

Definition at line 105 of file `stl_function.h`.

**5.115.2.2** `typedef bool std::unary_function< argument_type , bool >::result_type [inherited]`

`result_type` is the return type

Definition at line 108 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

### 5.116 `__gnu_parallel::_DRandomShufflingGlobalData<_RAIter>` Struct Template Reference

Data known to every thread participating in [\\_\\_gnu\\_parallel::\\_parallel\\_random\\_shuffle\(\)](#).

#### Public Types

- `typedef _TraitsType::difference_type _DifferenceType`

- `typedef std::iterator_traits<_RAIter> _TraitsType`
- `typedef _TraitsType::value_type _ValueType`

#### Public Member Functions

- [`\_DRandomShufflingGlobalData`](#) (`_RAIter &__source`)

#### Public Attributes

- [`\_ThreadIndex`](#) \* [`\_M\_bin\_proc`](#)
- `_DifferenceType` \*\* [`\_M\_dist`](#)
- `int` [`\_M\_num\_bins`](#)
- `int` [`\_M\_num\_bits`](#)
- `_RAIter &` [`\_M\_source`](#)
- `_DifferenceType` \* [`\_M\_starts`](#)
- `_ValueType` \*\* [`\_M\_temporaries`](#)

#### 5.116.1 Detailed Description

`template<typename _RAIter> struct __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>`

Data known to every thread participating in [`\_\_gnu\_parallel::\_\_parallel\_random\_shuffle\(\)`](#).

Definition at line 52 of file `random_shuffle.h`.

#### 5.116.2 Constructor & Destructor Documentation

**5.116.2.1** `template<typename _RAIter> __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::__DRandomShufflingGlobalData ( _RAIter & __source )`  
`[inline]`

Constructor.

Definition at line 83 of file `random_shuffle.h`.

### 5.116.3 Member Data Documentation

#### 5.116.3.1 `template<typename _RAIter> _ThreadIndex* __gnu_parallel::_DRandomShufflingGlobalData<_RAIter >::_M_bin_proc`

Number of the thread that will further process the corresponding bin.

Definition at line 74 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`.

#### 5.116.3.2 `template<typename _RAIter> _DifferenceType** __gnu_parallel::_DRandomShufflingGlobalData<_RAIter >::_M_dist`

Two-dimensional array to hold the thread-bin distribution.

Dimensions `( _M_num_threads + 1 ) __x ( _M_num_bins + 1 )`.

Definition at line 67 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs_pu()`.

#### 5.116.3.3 `template<typename _RAIter> int __gnu_ parallel::_DRandomShufflingGlobalData<_RAIter >::_M_num_bins`

Number of bins to distribute to.

Definition at line 77 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs_pu()`.

#### 5.116.3.4 `template<typename _RAIter> int __gnu_ parallel::_DRandomShufflingGlobalData<_RAIter >::_M_num_bits`

Number of bits needed to address the bins.

### 5.117 `__gnu_parallel::DRSSorterPU<_RAIter, _RandomNumberGenerator>` Struct Template Reference 1210

---

Definition at line 80 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs_pu()`.

#### 5.116.3.5 `template<typename _RAIter> _RAIter& __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::__M_source`

Begin iterator of the `__source`.

Definition at line 59 of file `random_shuffle.h`.

#### 5.116.3.6 `template<typename _RAIter> _DifferenceType* __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::__M_starts`

Start indexes of the threads' `__chunks`.

Definition at line 70 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs_pu()`.

#### 5.116.3.7 `template<typename _RAIter> _ValueType** __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::__M_temporaries`

Temporary arrays for each thread.

Definition at line 62 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`.

The documentation for this struct was generated from the following file:

- [random\\_shuffle.h](#)

### 5.117 `__gnu_parallel::DRSSorterPU<_RAIter, _RandomNumberGenerator>` Struct Template Reference

Local data for a thread participating in `__gnu_parallel::__parallel_random_shuffle()`.

## Public Attributes

- [\\_BinIndex \\_\\_bins\\_end](#)
- [\\_BinIndex \\_M\\_bins\\_begin](#)
- [int \\_M\\_num\\_threads](#)
- [\\_DRandomShufflingGlobalData<\\_RAIter> \\* \\_M\\_sd](#)
- [uint32\\_t \\_M\\_seed](#)

### 5.117.1 Detailed Description

`template<typename _RAIter, typename _RandomNumberGenerator> struct __gnu_parallel::DRSSorterPU<_RAIter, _RandomNumberGenerator>`

Local data for a thread participating in [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\(\)](#).

Definition at line 91 of file `random_shuffle.h`.

### 5.117.2 Member Data Documentation

**5.117.2.1** `template<typename _RAIter, typename _RandomNumberGenerator> _BinIndex __gnu_parallel::DRSSorterPU<_RAIter, _RandomNumberGenerator>::__bins_end`

End index for bins taken care of by this thread.

Definition at line 100 of file `random_shuffle.h`.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\\_drs\(\)](#).

**5.117.2.2** `template<typename _RAIter, typename _RandomNumberGenerator> _BinIndex __gnu_parallel::DRSSorterPU<_RAIter, _RandomNumberGenerator>::__M_bins_begin`

Begin index for bins taken care of by this thread.

Definition at line 97 of file `random_shuffle.h`.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\\_drs\(\)](#).

**5.117.2.3** `template<typename _RAIter, typename  
_RandomNumberGenerator> int __gnu_parallel::_DRSSorterPU<  
_RAIter, _RandomNumberGenerator >::_M_num_threads`

Number of threads participating in total.

Definition at line 94 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs_pu()`.

**5.117.2.4** `template<typename _RAIter, typename  
_RandomNumberGenerator> _DRandomShufflingGlobalData<_  
RAIter>* __gnu_parallel::_DRSSorterPU< _RAIter,  
_RandomNumberGenerator >::_M_sd`

Pointer to global data.

Definition at line 106 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs_pu()`.

**5.117.2.5** `template<typename _RAIter, typename  
_RandomNumberGenerator> uint32_t __gnu_parallel::_  
DRSSorterPU< _RAIter, _RandomNumberGenerator  
>::_M_seed`

Random `_M_seed` for this thread.

Definition at line 103 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs_pu()`.

The documentation for this struct was generated from the following file:

- [random\\_shuffle.h](#)

## 5.118 `__gnu_parallel::_DummyReduct` Struct Reference

Reduction function doing nothing.

## Public Member Functions

- `bool operator()` (`bool`, `bool`) `const`

### 5.118.1 Detailed Description

Reduction function doing nothing.

Definition at line 298 of file `for_each_selectors.h`.

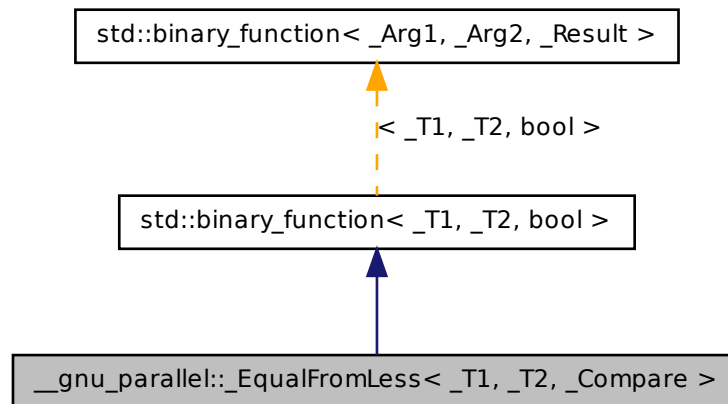
The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 5.119 `__gnu_parallel::_EqualFromLess< _T1, _T2, _Compare >` Class Template Reference

Constructs predicate for equality from strict weak ordering predicate.

Inheritance diagram for `__gnu_parallel::_EqualFromLess< _T1, _T2, _Compare >`:



## Public Types

- `typedef _T1 first_argument_type`



- typedef bool [result\\_type](#)
- typedef \_T2 [second\\_argument\\_type](#)

#### **Public Member Functions**

- **\_EqualFromLess** (\_Compare &\_\_comp)
- bool **operator()** (const \_T1 &\_\_a, const \_T2 &\_\_b)

#### **5.119.1 Detailed Description**

**template<typename \_T1, typename \_T2, typename \_Compare> class \_\_gnu\_parallel::\_EqualFromLess< \_T1, \_T2, \_Compare >**

Constructs predicate for equality from strict weak ordering predicate.

Definition at line 157 of file parallel/base.h.

#### **5.119.2 Member Typedef Documentation**

**5.119.2.1 typedef \_T1 std::binary\_function< \_T1 , \_T2 , bool >::first\_argument\_type [inherited]**

`first_argument_type` is the type of the first argument

Definition at line 118 of file stl\_function.h.

**5.119.2.2 typedef bool std::binary\_function< \_T1 , \_T2 , bool >::result\_type [inherited]**

`result_type` is the return type

Definition at line 124 of file stl\_function.h.

**5.119.2.3 typedef \_T2 std::binary\_function< \_T1 , \_T2 , bool >::second\_argument\_type [inherited]**

`second_argument_type` is the type of the second argument

Definition at line 121 of file stl\_function.h.

## 5.120 `__gnu_parallel::_EqualTo< _T1, _T2 >` Struct Template Reference 1215

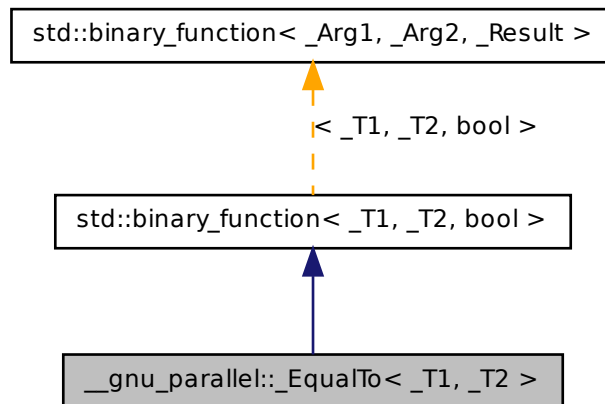
The documentation for this class was generated from the following file:

- [parallel/base.h](#)

## 5.120 `__gnu_parallel::_EqualTo< _T1, _T2 >` Struct Template Reference

Similar to [std::equal\\_to](#), but allows two different types.

Inheritance diagram for `__gnu_parallel::_EqualTo< _T1, _T2 >`:



### Public Types

- typedef `_T1` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_T2` [second\\_argument\\_type](#)

### Public Member Functions

- `bool operator() (const _T1 &__t1, const _T2 &__t2) const`

#### 5.120.1 Detailed Description

`template<typename _T1, typename _T2> struct __gnu_parallel::_EqualTo<_T1, _T2>`

Similar to `std::equal_to`, but allows two different types.

Definition at line 244 of file `parallel/base.h`.

#### 5.120.2 Member Typedef Documentation

5.120.2.1 `typedef _T1 std::binary_function<_T1, _T2, bool>::first_argument_type [inherited]`

`first_argument_type` is the type of the first argument

Definition at line 118 of file `stl_function.h`.

5.120.2.2 `typedef bool std::binary_function<_T1, _T2, bool>::result_type [inherited]`

`result_type` is the return type

Definition at line 124 of file `stl_function.h`.

5.120.2.3 `typedef _T2 std::binary_function<_T1, _T2, bool>::second_argument_type [inherited]`

`second_argument_type` is the type of the second argument

Definition at line 121 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [parallel/base.h](#)

### 5.121 `__gnu_parallel::_GuardedIterator<_RAIter, _Compare>` Class Template Reference

`_Iterator` wrapper supporting an implicit supremum at the end of the sequence, dominating all comparisons.

## Public Member Functions

- `_GuardedIterator` (`_RAIter __begin`, `_RAIter __end`, `_Compare &__comp`)
- `operator _RAIter` ()
- `std::iterator_traits<_RAIter>::value_type & operator*` ()
- `_GuardedIterator<_RAIter, _Compare> & operator++` ()

## Friends

- `bool operator<` (`_GuardedIterator<_RAIter, _Compare> &__bi1`, `_GuardedIterator<_RAIter, _Compare> &__bi2`)
- `bool operator<=` (`_GuardedIterator<_RAIter, _Compare> &__bi1`, `_GuardedIterator<_RAIter, _Compare> &__bi2`)

### 5.121.1 Detailed Description

`template<typename _RAIter, typename _Compare> class __gnu_parallel::_GuardedIterator<_RAIter, _Compare>`

`_Iterator` wrapper supporting an implicit supremum at the end of the sequence, dominating all comparisons. The implicit supremum comes with a performance cost.

Deriving from `_RAIter` is not possible since `_RAIter` need not be a class.

Definition at line 66 of file `multiway_merge.h`.

### 5.121.2 Constructor & Destructor Documentation

**5.121.2.1** `template<typename _RAIter, typename _Compare> __gnu_parallel::_GuardedIterator<_RAIter, _Compare>::_GuardedIterator ( _RAIter __begin, _RAIter __end, _Compare & __comp ) [inline]`

Constructor. Sets iterator to beginning of sequence.

#### Parameters

- `__begin` Begin iterator of sequence.
- `__end` End iterator of sequence.
- `__comp` Comparator provided for associated overloaded compare operators.

Definition at line 84 of file `multiway_merge.h`.

### 5.121.3 Member Function Documentation

**5.121.3.1** `template<typename _RAIter, typename _Compare>`  
`__gnu_parallel::_GuardedIterator<_RAIter, _Compare>::operator`  
`_RAIter ( ) [inline]`

Convert to wrapped iterator.

#### Returns

Wrapped iterator.

Definition at line 105 of file `multiway_merge.h`.

**5.121.3.2** `template<typename _RAIter, typename _Compare`  
`> std::iterator_traits<_RAIter>::value_type&`  
`__gnu_parallel::_GuardedIterator<_RAIter, _Compare`  
`>::operator* ( ) [inline]`

Dereference operator.

#### Returns

Referenced element.

Definition at line 100 of file `multiway_merge.h`.

**5.121.3.3** `template<typename _RAIter, typename _Compare`  
`> _GuardedIterator<_RAIter, _Compare>&`  
`__gnu_parallel::_GuardedIterator<_RAIter, _Compare`  
`>::operator++ ( ) [inline]`

Pre-increment operator.

#### Returns

This.

Definition at line 91 of file `multiway_merge.h`.

#### 5.121.4 Friends And Related Function Documentation

**5.121.4.1** `template<typename _RAIter, typename _Compare> bool  
operator< ( _GuardedIterator<_RAIter, _Compare> & __bi1,  
_GuardedIterator<_RAIter, _Compare> & __bi2 ) [friend]`

Compare two elements referenced by guarded iterators.

##### Parameters

`__bi1` First iterator.  
`__bi2` Second iterator.

##### Returns

`true` if less.

Definition at line 113 of file `multiway_merge.h`.

**5.121.4.2** `template<typename _RAIter, typename _Compare> bool  
operator<= ( _GuardedIterator<_RAIter, _Compare> & __bi1,  
_GuardedIterator<_RAIter, _Compare> & __bi2 ) [friend]`

Compare two elements referenced by guarded iterators.

##### Parameters

`__bi1` First iterator.  
`__bi2` Second iterator.

##### Returns

`True` if less equal.

Definition at line 128 of file `multiway_merge.h`.

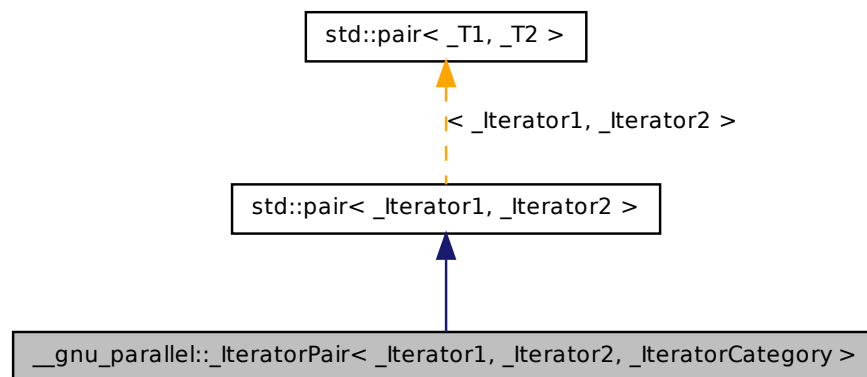
The documentation for this class was generated from the following file:

- [multiway\\_merge.h](#)

## 5.122 `__gnu_parallel::IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >` Class Template Reference

A pair of iterators. The usual iterator operations are applied to both child iterators.

Inheritance diagram for `__gnu_parallel::IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >`:



### Public Types

- `typedef std::iterator_traits< _Iterator1 > TraitsType`
- `typedef TraitsType::difference_type difference_type`
- `typedef _Iterator1 first_type`
- `typedef _IteratorCategory iterator_category`
- `typedef _IteratorPair * pointer`
- `typedef _IteratorPair & reference`
- `typedef _Iterator2 second_type`
- `typedef void value_type`

### Public Member Functions

- `_IteratorPair` (`const _Iterator1 &__first`, `const _Iterator2 &__second`)
- `operator _Iterator2` () `const`
- `_IteratorPair operator+` (`difference_type __delta`) `const`

- `const _IteratorPair operator++` (int)
- `_IteratorPair & operator++` ()
- `difference_type operator-` (const `_IteratorPair` &\_\_other) const
- `const _IteratorPair operator--` (int)
- `_IteratorPair & operator--` ()
- `_IteratorPair & operator=` (const `_IteratorPair` &\_\_other)
- `void swap` (pair &\_\_p)

#### Public Attributes

- `_Iterator1` `first`
- `_Iterator2` `second`

#### 5.122.1 Detailed Description

`template<typename _Iterator1, typename _Iterator2, typename _IteratorCategory> class __gnu_parallel::IteratorPair<_Iterator1, _Iterator2, _IteratorCategory>`

A pair of iterators. The usual iterator operations are applied to both child iterators.

Definition at line 45 of file `iterator.h`.

#### 5.122.2 Member Typedef Documentation

5.122.2.1 `typedef _Iterator2 std::pair<_Iterator1, _Iterator2>::second_type` `[inherited]`

`first_type` is the first bound type

Definition at line 90 of file `stl_pair.h`.

#### 5.122.3 Member Data Documentation

5.122.3.1 `_Iterator1 std::pair<_Iterator1, _Iterator2>::first` `[inherited]`

`second_type` is the second bound type

Definition at line 92 of file `stl_pair.h`.



5.122.3.2 `_Iterator2 std::pair<_Iterator1, _Iterator2>::second`  
[inherited]

`first` is a copy of the first object

Definition at line 93 of file `stl_pair.h`.

The documentation for this class was generated from the following file:

- [iterator.h](#)

5.123 `__gnu_parallel::IteratorTriple<_Iterator1, _Iterator2, _Iterator3, _IteratorCategory>` Class Template Reference

A triple of iterators. The usual iterator operations are applied to all three child iterators.

**Public Types**

- `typedef std::iterator_traits<_Iterator1>::difference_type` **difference\_type**
- `typedef _IteratorCategory` **iterator\_category**
- `typedef _IteratorTriple *` **pointer**
- `typedef _IteratorTriple &` **reference**
- `typedef void` **value\_type**

**Public Member Functions**

- `_IteratorTriple` (`const _Iterator1 &__first`, `const _Iterator2 &__second`, `const _Iterator3 &__third`)
- `operator _Iterator3` () `const`
- `_IteratorTriple operator+` (`difference_type __delta`) `const`
- `const _IteratorTriple operator++` (`int`)
- `_IteratorTriple & operator++` ()
- `difference_type operator-` (`const _IteratorTriple &__other`) `const`
- `_IteratorTriple & operator--` ()
- `const _IteratorTriple operator--` (`int`)
- `_IteratorTriple & operator=` (`const _IteratorTriple &__other`)

**Public Attributes**

- `_Iterator1` **\_M\_first**
- `_Iterator2` **\_M\_second**
- `_Iterator3` **\_M\_third**

## 5.124 `__gnu_parallel::__Job<_DifferenceTp>` Struct Template Reference 1223

### 5.123.1 Detailed Description

```
template<typename _Iterator1, typename _Iterator2, typename _Iterator3, type-
name _IteratorCategory> class __gnu_parallel::_IteratorTriple< _Iterator1, _-
Iterator2, _Iterator3, _IteratorCategory >
```

A triple of iterators. The usual iterator operations are applied to all three child iterators.  
Definition at line 120 of file `iterator.h`.

The documentation for this class was generated from the following file:

- [iterator.h](#)

## 5.124 `__gnu_parallel::__Job<_DifferenceTp>` Struct Template Reference

One `__job` for a certain thread.

### Public Types

- typedef `_DifferenceTp` `_DifferenceType`

### Public Attributes

- volatile `_DifferenceType` `_M_first`
- volatile `_DifferenceType` `_M_last`
- volatile `_DifferenceType` `_M_load`

### 5.124.1 Detailed Description

```
template<typename _DifferenceTp> struct __gnu_parallel::__Job< _-
DifferenceTp >
```

One `__job` for a certain thread.

Definition at line 54 of file `workstealing.h`.

### 5.124.2 Member Data Documentation

5.124.2.1 `template<typename _DifferenceTp> volatile _DifferenceType`  
`__gnu_parallel::__Job<_DifferenceTp>::__M_first`

First element.

Changed by owning and stealing thread. By stealing thread, always incremented.

Definition at line 62 of file `workstealing.h`.

Referenced by `__gnu_parallel::__for_each_template_random_access_workstealing()`.

**5.124.2.2 `template<typename _DifferenceTp> volatile _DifferenceType  
__gnu_parallel::_Job<_DifferenceTp>::__M_last`**

Last element.

Changed by owning thread only.

Definition at line 67 of file `workstealing.h`.

Referenced by `__gnu_parallel::__for_each_template_random_access_workstealing()`.

**5.124.2.3 `template<typename _DifferenceTp> volatile _DifferenceType  
__gnu_parallel::_Job<_DifferenceTp>::__M_load`**

Number of elements, i.e. `_M_last-_M_first+1`.

Changed by owning thread only.

Definition at line 72 of file `workstealing.h`.

Referenced by `__gnu_parallel::__for_each_template_random_access_workstealing()`.

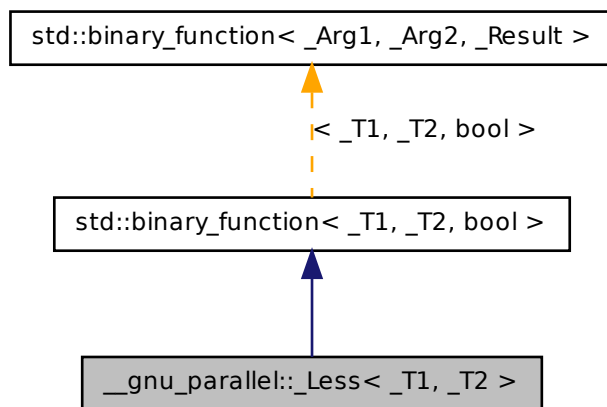
The documentation for this struct was generated from the following file:

- [workstealing.h](#)

**5.125 `__gnu_parallel::__Less<_T1, _T2>` Struct Template Reference**

Similar to [std::less](#), but allows two different types.

Inheritance diagram for `__gnu_parallel::_Less<_T1, _T2>`:



### Public Types

- typedef `_T1` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_T2` [second\\_argument\\_type](#)

### Public Member Functions

- `bool operator() (const _T1 &__t1, const _T2 &__t2) const`
- `bool operator() (const _T2 &__t2, const _T1 &__t1) const`

#### 5.125.1 Detailed Description

`template<typename _T1, typename _T2> struct __gnu_parallel::_Less<_T1, _T2>`

Similar to [std::less](#), but allows two different types.

Definition at line 252 of file `parallel/base.h`.

### 5.125.2 Member Typedef Documentation

**5.125.2.1** `typedef _T1 std::binary_function<_T1, _T2, bool >::first_argument_type [inherited]`

`first_argument_type` is the type of the first argument

Definition at line 118 of file `stl_function.h`.

**5.125.2.2** `typedef bool std::binary_function<_T1, _T2, bool >::result_type [inherited]`

`result_type` is the return type

Definition at line 124 of file `stl_function.h`.

**5.125.2.3** `typedef _T2 std::binary_function<_T1, _T2, bool >::second_argument_type [inherited]`

`second_argument_type` is the type of the second argument

Definition at line 121 of file `stl_function.h`.

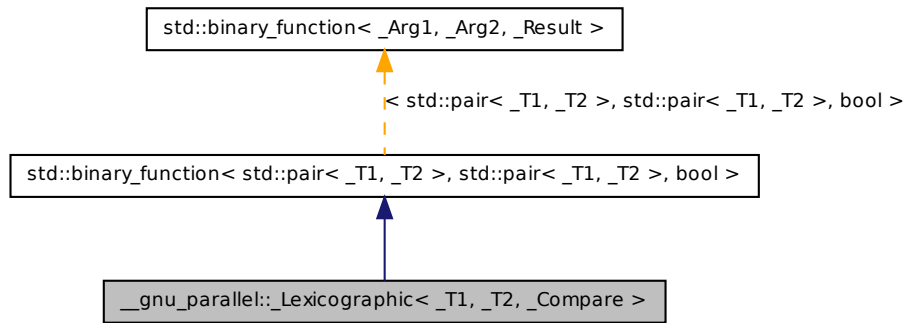
The documentation for this struct was generated from the following file:

- [parallel/base.h](#)

## 5.126 `__gnu_parallel::_Lexicographic<_T1, _T2, _Compare>` Class Template Reference

Compare \_\_a pair of types lexicographically, ascending.

Inheritance diagram for `__gnu_parallel::_Lexicographic<_T1, _T2, _Compare>`:



### Public Types

- typedef `std::pair<_T1, _T2>` `first_argument_type`
- typedef `bool` `result_type`
- typedef `std::pair<_T1, _T2>` `second_argument_type`

### Public Member Functions

- `_Lexicographic` (`_Compare` &`__comp`)
- `bool operator()` (`const std::pair<_T1, _T2>` &`__p1`, `const std::pair<_T1, _T2>` &`__p2`) `const`

#### 5.126.1 Detailed Description

`template<typename _T1, typename _T2, typename _Compare> class __gnu_parallel::_Lexicographic<_T1, _T2, _Compare>`

Compare \_\_a pair of types lexicographically, ascending.

Definition at line 55 of file `multiseq_selection.h`.

### 5.126.2 Member Typedef Documentation

**5.126.2.1** `typedef std::pair< _T1, _T2 > std::binary_function< std::pair< _T1, _T2 >, std::pair< _T1, _T2 >, bool >::first_argument_type`  
[`inherited`]

`first_argument_type` is the type of the first argument

Definition at line 118 of file `stl_function.h`.

**5.126.2.2** `typedef bool std::binary_function< std::pair< _T1, _T2 >, std::pair< _T1, _T2 >, bool >::result_type` [`inherited`]

`result_type` is the return type

Definition at line 124 of file `stl_function.h`.

**5.126.2.3** `typedef std::pair< _T1, _T2 > std::binary_function< std::pair< _T1, _T2 >, std::pair< _T1, _T2 >, bool >::second_argument_type`  
[`inherited`]

`second_argument_type` is the type of the second argument

Definition at line 121 of file `stl_function.h`.

The documentation for this class was generated from the following file:

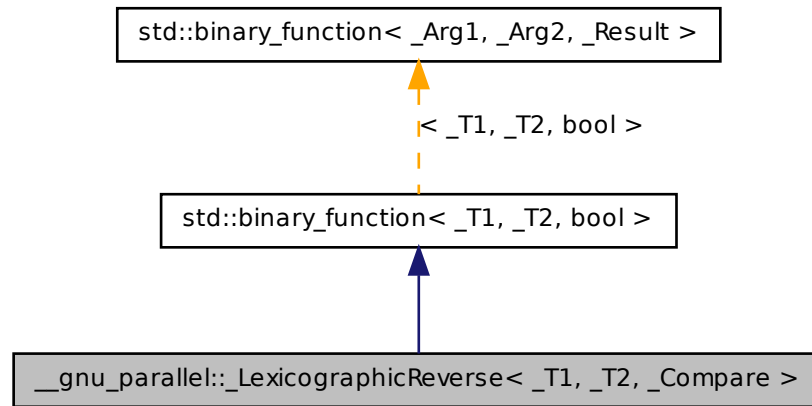
- [multiseq\\_selection.h](#)

## 5.127 `__gnu_parallel::_LexicographicReverse< _T1, _T2, _Compare >` Class Template Reference

Compare \_\_a pair of types lexicographically, descending.

Inheritance diagram for `__gnu_parallel::_LexicographicReverse< _T1, _T2, _-`

Compare >:



### Public Types

- `typedef _T1 first_argument_type`
- `typedef bool result_type`
- `typedef _T2 second_argument_type`

### Public Member Functions

- `_LexicographicReverse` (`_Compare &__comp`)
- `bool operator()` (`const std::pair<_T1, _T2> &__p1, const std::pair<_T1, _T2> &__p2`) `const`

#### 5.127.1 Detailed Description

`template<typename _T1, typename _T2, typename _Compare> class __gnu_parallel::_LexicographicReverse<_T1, _T2, _Compare>`

Compare \_\_a pair of types lexicographically, descending.

Definition at line 82 of file `multiseq_selection.h`.



### 5.127.2 Member Typedef Documentation

5.127.2.1 `typedef _T1 std::binary_function< _T1 , _T2 , bool  
>::first_argument_type [inherited]`

`first_argument_type` is the type of the first argument

Definition at line 118 of file `stl_function.h`.

5.127.2.2 `typedef bool std::binary_function< _T1 , _T2 , bool >::result_type  
[inherited]`

`result_type` is the return type

Definition at line 124 of file `stl_function.h`.

5.127.2.3 `typedef _T2 std::binary_function< _T1 , _T2 , bool  
>::second_argument_type [inherited]`

`second_argument_type` is the type of the second argument

Definition at line 121 of file `stl_function.h`.

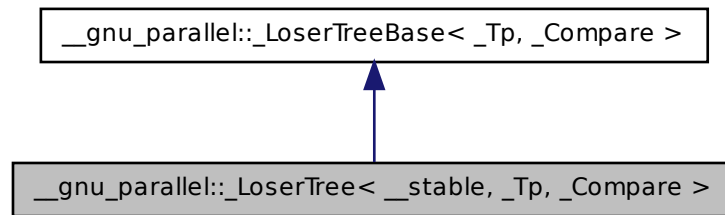
The documentation for this class was generated from the following file:

- [multiseq\\_selection.h](#)

## 5.128 `__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >` Class Template Reference

Stable [\\_LoserTree](#) variant.

Inheritance diagram for `__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >`:



### Public Member Functions

- **`_LoserTree`** (unsigned int `__k`, `_Compare` `__comp`)
- void `__delete_min_insert` (`_Tp` `__key`, bool `__sup`)
- int `__get_min_source` ()
- void `__init` ()
- unsigned int `__init_winner` (unsigned int `__root`)
- void `__insert_start` (const `_Tp` &`__key`, int `__source`, bool `__sup`)

### Protected Attributes

- `_Compare` `_M_comp`
- bool `_M_first_insert`
- unsigned int `_M_ik`
- unsigned int `_M_k`
- unsigned int `_M_log_k`
- `_Loser` \* `_M_losers`
- unsigned int `_M_offset`

#### 5.128.1 Detailed Description

`template<bool __stable, typename _Tp, typename _Compare> class __gnu_parallel::_LoserTree< __stable, _Tp, _Compare >`

Stable `_LoserTree` variant. Provides the stable implementations of `insert_start`, `__init_winner`, `__init` and `__delete_min_insert`.

Unstable variant is done using partial specialisation below.

Definition at line 165 of file `losertree.h`.

### 5.128.2 Member Function Documentation

**5.128.2.1** `template<bool __stable, typename _Tp , typename _Compare  
> void __gnu_parallel::_LoserTree< __stable, _Tp, _Compare  
>::__delete_min_insert( _Tp __key, bool __sup ) [inline]`

Delete the smallest element and insert a new element from the previously smallest element's sequence.

This implementation is stable.

Definition at line 217 of file `losertree.h`.

**5.128.2.2** `template<typename _Tp , typename _Compare >  
int __gnu_parallel::_LoserTreeBase< _Tp, _Compare  
>::__get_min_source( ) [inline, inherited]`

#### Returns

the index of the sequence with the smallest element.

Definition at line 151 of file `losertree.h`.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`.

**5.128.2.3** `template<typename _Tp , typename _Compare > void  
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start  
( const _Tp & __key, int __source, bool __sup ) [inline,  
inherited]`

Initializes the sequence "`_M_source`" with the element "`__key`".

#### Parameters

`__key` the element to insert

`__source` \_\_index of the \_\_source \_\_sequence

`__sup` flag that determines whether the value to insert is an explicit `__supremum`.

Definition at line 130 of file `losertree.h`.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_key`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_sup`.

### 5.128.3 Member Data Documentation

#### 5.128.3.1 `template<typename _Tp, typename _Compare > _Compare __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_comp` [protected, inherited]

`_Compare` to use.

Definition at line 78 of file `losertree.h`.

#### 5.128.3.2 `template<typename _Tp, typename _Compare > bool __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert` [protected, inherited]

State flag that determines whether the `_LoserTree` is empty.

Only used for building the `_LoserTree`.

Definition at line 85 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start()`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

#### 5.128.3.3 `template<typename _Tp, typename _Compare > unsigned int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k` [protected, inherited]

`log_2{ _M_k }`

Definition at line 72 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

## 5.129 `__gnu_parallel::_LoserTree< false, _Tp, _Compare >` Class Template Reference 1234

---

5.128.3.4 `template<typename _Tp , typename _Compare > _Loser*`  
`__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__M_losers`  
`[protected, inherited]`

[\\_LoserTree](#) \_\_elements.

Definition at line 75 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__get_min_source()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::~~LoserTreeBase()`.

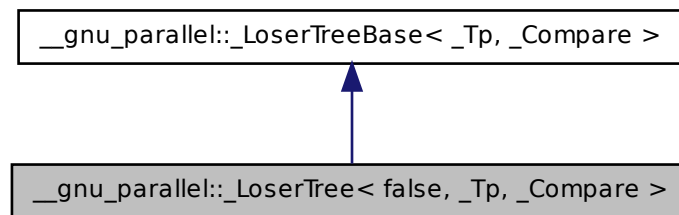
The documentation for this class was generated from the following file:

- [losertree.h](#)

## 5.129 `__gnu_parallel::_LoserTree< false, _Tp, _Compare >` Class Template Reference

Unstable [\\_LoserTree](#) variant.

Inheritance diagram for `__gnu_parallel::_LoserTree< false, _Tp, _Compare >`:



### Public Member Functions

- `_LoserTree` (unsigned int \_\_k, \_Compare \_\_comp)
- void [\\_\\_delete\\_min\\_insert](#) (\_Tp \_\_key, bool \_\_sup)
- int [\\_\\_get\\_min\\_source](#) ()

- void `__init` ()
- unsigned int `__init_winner` (unsigned int `__root`)
- void `__insert_start` (const `_Tp` &`__key`, int `__source`, bool `__sup`)

#### Protected Attributes

- `_Compare` `_M_comp`
- bool `_M_first_insert`
- unsigned int `_M_ik`
- unsigned int `_M_k`
- unsigned int `_M_log_k`
- `_Loser` \* `_M_losers`
- unsigned int `_M_offset`

#### 5.129.1 Detailed Description

`template<typename _Tp, typename _Compare> class __gnu_parallel::_LoserTree< false, _Tp, _Compare >`

Unstable `_LoserTree` variant. Stability (non-stable here) is selected with partial specialization.

Definition at line 256 of file `losertree.h`.

#### 5.129.2 Member Function Documentation

**5.129.2.1** `template<typename _Tp , typename _Compare > void __gnu_parallel::_LoserTree< false, _Tp, _Compare >::__delete_min_insert ( _Tp __key, bool __sup ) [inline]`

Delete the `_M_key` smallest element and insert the element `__key` instead.

##### Parameters

- `__key` the `_M_key` to insert
- `__sup` true iff `__key` is an explicitly marked supremum

Definition at line 318 of file `losertree.h`.

**5.129.2.2** `template<typename _Tp , typename _Compare > int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__get_min_source ( ) [inline, inherited]`

**5.129 \_\_gnu\_parallel::\_LoserTree< false, \_Tp, \_Compare > Class Template Reference** **1236**

---

**Returns**

the index of the sequence with the smallest element.

Definition at line 151 of file losertree.h.

References \_\_gnu\_parallel::\_LoserTreeBase< \_Tp, \_Compare >::\_M\_losers, and \_\_gnu\_parallel::\_LoserTreeBase< \_Tp, \_Compare >::\_Loser::\_M\_source.

**5.129.2.3 template<typename \_Tp , typename \_Compare > unsigned int  
\_\_gnu\_parallel::\_LoserTree< false, \_Tp, \_Compare >::\_\_init\_winner  
( unsigned int \_\_root ) [inline]**

Computes the winner of the competition at position "\_\_root".

Called recursively (starting at 0) to build the initial tree.

**Parameters**

**\_\_root** \_\_index of the "game" to start.

Definition at line 278 of file losertree.h.

**5.129.2.4 template<typename \_Tp , typename \_Compare > void  
\_\_gnu\_parallel::\_LoserTreeBase< \_Tp, \_Compare >::\_\_insert\_start  
( const \_Tp & \_\_key, int \_\_source, bool \_\_sup ) [inline,  
inherited]**

Initializes the sequence "\_M\_source" with the element "\_\_key".

**Parameters**

**\_\_key** the element to insert

**\_\_source** \_\_index of the \_\_source \_\_sequence

**\_\_sup** flag that determines whether the value to insert is an explicit \_\_supremum.

Definition at line 130 of file losertree.h.

References \_\_gnu\_parallel::\_LoserTreeBase< \_Tp, \_Compare >::\_M\_first\_insert, \_\_gnu\_parallel::\_LoserTreeBase< \_Tp, \_Compare >::\_Loser::\_M\_key, \_\_gnu\_parallel::\_LoserTreeBase< \_Tp, \_Compare >::\_M\_losers, \_\_gnu\_parallel::\_LoserTreeBase< \_Tp, \_Compare >::\_Loser::\_M\_source, and \_\_gnu\_parallel::\_LoserTreeBase< \_Tp, \_Compare >::\_Loser::\_M\_sup.

### 5.129.3 Member Data Documentation

**5.129.3.1** `template<typename _Tp , typename _Compare > _Compare  
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_comp  
[protected, inherited]`

`_Compare` to use.

Definition at line 78 of file `losertree.h`.

**5.129.3.2** `template<typename _Tp , typename _Compare >  
bool __gnu_parallel::_LoserTreeBase< _Tp, _Compare  
>::_M_first_insert [protected, inherited]`

State flag that determines whether the `_LoserTree` is empty.

Only used for building the `_LoserTree`.

Definition at line 85 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start()`,  
and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

**5.129.3.3** `template<typename _Tp , typename _Compare > unsigned int  
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k  
[protected, inherited]`

`log_2{_M_k}`

Definition at line 72 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

**5.129.3.4** `template<typename _Tp , typename _Compare > _Loser*  
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers  
[protected, inherited]`

`_LoserTree` \_\_elements.

Definition at line 75 of file `losertree.h`.



### 5.130 `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >` Class Template Reference 1238

---

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__get_min_source()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::~~LoserTreeBase()`.

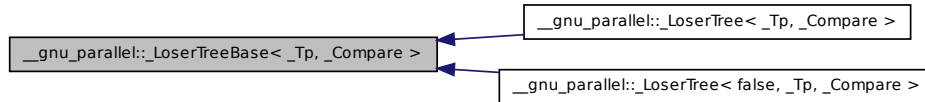
The documentation for this class was generated from the following file:

- [losertree.h](#)

### 5.130 `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >` Class Template Reference

Guarded loser/tournament tree.

Inheritance diagram for `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >`:



#### Classes

- struct [\\_Loser](#)  
*Internal representation of a [\\_LoserTree](#) element.*

#### Public Member Functions

- [\\_LoserTreeBase](#) (unsigned int \_\_k, \_Compare \_\_comp)
- [~LoserTreeBase](#) ()
- int [\\_\\_get\\_min\\_source](#) ()
- void [\\_\\_insert\\_start](#) (const \_Tp &\_\_key, int \_\_source, bool \_\_sup)

#### Protected Attributes

- \_Compare [\\_M\\_comp](#)
- bool [\\_M\\_first\\_insert](#)
- unsigned int [\\_M\\_ik](#)

- unsigned int `_M_k`
- unsigned int `_M_log_k`
- `_Loser * _M_losers`
- unsigned int `_M_offset`

### 5.130.1 Detailed Description

`template<typename _Tp, typename _Compare> class __gnu_parallel::_LoserTreeBase< _Tp, _Compare >`

Guarded loser/tournament tree. The smallest element is at the top.

Guarding is done explicitly through one flag `_M_sup` per element, `inf` is not needed due to a better initialization routine. This is a well-performing variant.

#### Parameters

- `_Tp` the element type
- `_Compare` the comparator to use, defaults to `std::less<_Tp>`

Definition at line 55 of file `losertree.h`.

### 5.130.2 Constructor & Destructor Documentation

**5.130.2.1** `template<typename _Tp , typename _Compare >  
__gnu_parallel::_LoserTreeBase< _Tp, _Compare  
>::_LoserTreeBase ( unsigned int __k, _Compare __comp )  
[inline]`

The constructor.

#### Parameters

- `__k` The number of sequences to merge.
- `__comp` The comparator to use.

Definition at line 94 of file `losertree.h`.

References `__gnu_parallel::__rd_log2()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`.

5.130.2.2 `template<typename _Tp , typename _Compare >`  
`__gnu_parallel::_LoserTreeBase< _Tp, _Compare`  
`>::~~LoserTreeBase ( ) [inline]`

The destructor.

Definition at line 118 of file `losertree.h`.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`.

### 5.130.3 Member Function Documentation

5.130.3.1 `template<typename _Tp , typename _Compare >`  
`int __gnu_parallel::_LoserTreeBase< _Tp, _Compare`  
`>::_get_min_source ( ) [inline]`

#### Returns

the index of the sequence with the smallest element.

Definition at line 151 of file `losertree.h`.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`.

5.130.3.2 `template<typename _Tp , typename _Compare > void`  
`__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start`  
`( const _Tp & __key, int __source, bool __sup ) [inline]`

Initializes the sequence "`_M_source`" with the element "`__key`".

#### Parameters

`__key` the element to insert

`__source` \_\_index of the `__source` \_\_sequence

`__sup` flag that determines whether the value to insert is an explicit `__supremum`.

Definition at line 130 of file `losertree.h`.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_key`,

### 5.130 `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >` Class Template Reference 1241

---

`__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_sup`.

#### 5.130.4 Member Data Documentation

##### 5.130.4.1 `template<typename _Tp, typename _Compare > _Compare` `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_comp` `[protected]`

`_Compare` to use.

Definition at line 78 of file `losertree.h`.

##### 5.130.4.2 `template<typename _Tp, typename _Compare >` `bool __gnu_parallel::_LoserTreeBase< _Tp, _Compare` `>::_M_first_insert [protected]`

State flag that determines whether the [\\_LoserTree](#) is empty.

Only used for building the [\\_LoserTree](#).

Definition at line 85 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start()`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

##### 5.130.4.3 `template<typename _Tp, typename _Compare > unsigned int` `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k` `[protected]`

`log_2{_M_k}`

Definition at line 72 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

##### 5.130.4.4 `template<typename _Tp, typename _Compare > _Loser*` `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers` `[protected]`

### 5.131 `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser` Struct Reference 1242

---

[\\_LoserTree](#) \_\_elements.

Definition at line 75 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__get_min_source()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::~~LoserTreeBase()`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

### 5.131 `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser` Struct Reference

Internal representation of a [\\_LoserTree](#) element.

#### Public Attributes

- `_Tp` [\\_M\\_key](#)
- `int` [\\_M\\_source](#)
- `bool` [\\_M\\_sup](#)

#### 5.131.1 Detailed Description

`template<typename _Tp, typename _Compare> struct __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser`

Internal representation of a [\\_LoserTree](#) element.

Definition at line 59 of file `losertree.h`.

#### 5.131.2 Member Data Documentation

**5.131.2.1** `template<typename _Tp , typename _Compare > _Tp __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_key`

`_M_key` of the element in the [\\_LoserTree](#).

Definition at line 66 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start()`.

## 5.132 `__gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >` Class Template Reference 1243

---

**5.131.2.2** `template<typename _Tp , typename _Compare >`  
`int __gnu_parallel::_LoserTreeBase< _Tp, _Compare`  
`>::_Loser::_M_source`

`__index` of the `__source` `__sequence`.

Definition at line 64 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__get_min_source()`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start()`.

**5.131.2.3** `template<typename _Tp , typename _Compare >`  
`bool __gnu_parallel::_LoserTreeBase< _Tp, _Compare`  
`>::_Loser::_M_sup`

flag, true iff this is a "maximum" `__sentinel`.

Definition at line 62 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start()`.

The documentation for this struct was generated from the following file:

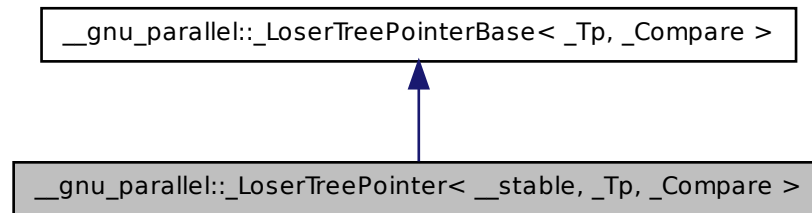
- [losertree.h](#)

## 5.132 `__gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >` Class Template Reference

Stable [\\_LoserTree](#) implementation.

Inheritance diagram for `__gnu_parallel::_LoserTreePointer< __stable, _Tp, _`

Compare >:



### Public Member Functions

- **\_LoserTreePointer** (unsigned int \_\_k, \_Compare \_\_comp=[std::less](#)< \_Tp >())
- void **\_\_delete\_min\_insert** (const \_Tp &\_\_key, bool \_\_sup)
- int **\_\_get\_min\_source** ()
- void **\_\_init** ()
- unsigned int **\_\_init\_winner** (unsigned int \_\_root)
- void **\_\_insert\_start** (const \_Tp &\_\_key, int \_\_source, bool \_\_sup)

### Protected Attributes

- \_Compare **\_M\_comp**
- unsigned int **\_M\_ik**
- unsigned int **\_M\_k**
- [\\_Loser](#) \* **\_M\_losers**
- unsigned int **\_M\_offset**

#### 5.132.1 Detailed Description

`template<bool __stable, typename _Tp, typename _Compare> class __gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >`

Stable [\\_LoserTree](#) implementation. The unstable variant is implemented using partial instantiation below.

Definition at line 403 of file losertree.h.

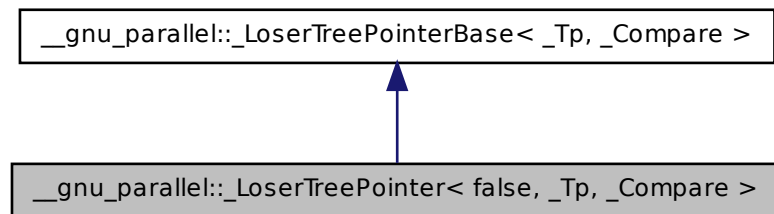
The documentation for this class was generated from the following file:

- [losertree.h](#)

### 5.133 `__gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >` Class Template Reference

Unstable [\\_LoserTree](#) implementation.

Inheritance diagram for `__gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >`:



#### Public Member Functions

- `_LoserTreePointer` (unsigned int \_\_k, \_Compare \_\_comp=[std::less](#)< \_Tp >())
- `void __delete_min_insert` (const \_Tp &\_\_key, bool \_\_sup)
- `int __get_min_source` ()
- `void __init` ()
- `unsigned int __init_winner` (unsigned int \_\_root)
- `void __insert_start` (const \_Tp &\_\_key, int \_\_source, bool \_\_sup)

#### Protected Attributes

- `_Compare _M_comp`
- `unsigned int _M_ik`
- `unsigned int _M_k`
- `\_Loser * _M_losers`
- `unsigned int _M_offset`



### 5.133.1 Detailed Description

`template<typename _Tp, typename _Compare> class __gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >`

Unstable [\\_LoserTree](#) implementation. The stable variant is above.

Definition at line 484 of file `losertree.h`.

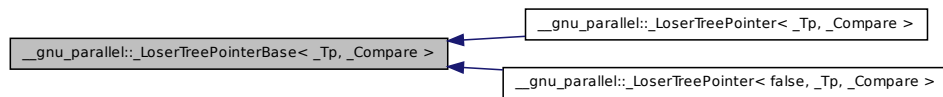
The documentation for this class was generated from the following file:

- [losertree.h](#)

## 5.134 `__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >` Class Template Reference

Base class of [\\_Loser](#) Tree implementation using pointers.

Inheritance diagram for `__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >`:



### Classes

- struct [\\_Loser](#)  
*Internal representation of [\\_LoserTree](#) \_\_elements.*

### Public Member Functions

- `_LoserTreePointerBase` (unsigned int \_\_k, \_Compare \_\_comp=[std::less](#)< \_Tp >())
- `int __get_min_source ()`
- `void __insert_start` (const \_Tp &\_\_key, int \_\_source, bool \_\_sup)

### Protected Attributes

- `_Compare _M_comp`

### 5.135 `__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser` Struct Reference 1247

---

- unsigned int `_M_ik`
- unsigned int `_M_k`
- [\\_Loser](#) \* `_M_losers`
- unsigned int `_M_offset`

#### 5.134.1 Detailed Description

**template<typename `_Tp`, typename `_Compare`> class `__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >`**

Base class of [\\_Loser](#) Tree implementation using pointers.

Definition at line 351 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

### 5.135 `__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser` Struct Reference

Internal representation of [\\_LoserTree](#) \_\_elements.

#### Public Attributes

- const `_Tp` \* `_M_keyp`
- int `_M_source`
- bool `_M_sup`

#### 5.135.1 Detailed Description

**template<typename `_Tp`, typename `_Compare`> struct `__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser`**

Internal representation of [\\_LoserTree](#) \_\_elements.

Definition at line 355 of file `losertree.h`.

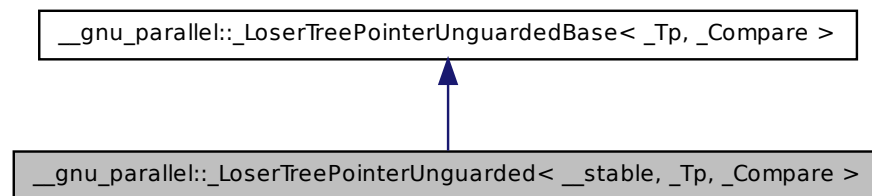
The documentation for this struct was generated from the following file:

- [losertree.h](#)

### 5.136 `__gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >` Class Template Reference

Stable unguarded [\\_LoserTree](#) variant storing pointers.

Inheritance diagram for `__gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >`:



#### Public Member Functions

- **`_LoserTreePointerUnguarded`** (unsigned int `__k`, const `_Tp` &`__sentinel`, `_Compare` `__comp`=`std::less< _Tp >()`)
- void **`__delete_min_insert`** (const `_Tp` &`__key`, bool `__sup`)
- int **`__get_min_source`** ()
- void **`__init`** ()
- unsigned int **`__init_winner`** (unsigned int `__root`)
- void **`__insert_start`** (const `_Tp` &`__key`, int `__source`, bool)

#### Protected Attributes

- `_Compare` **`_M_comp`**
- unsigned int **`_M_ik`**
- unsigned int **`_M_k`**
- `_Loser` \* **`_M_losers`**
- unsigned int **`_M_offset`**

### 5.136.1 Detailed Description

`template<bool __stable, typename _Tp, typename _Compare> class __gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >`

Stable unguarded [\\_LoserTree](#) variant storing pointers. Unstable variant is implemented below using partial specialization.

Definition at line 871 of file `losertree.h`.

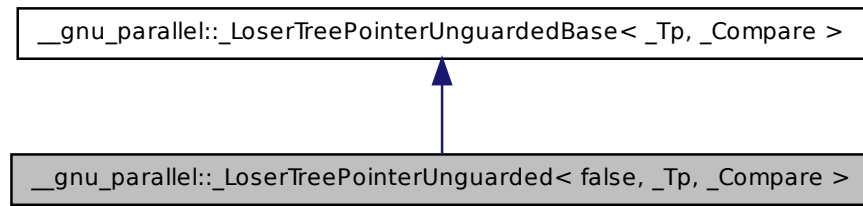
The documentation for this class was generated from the following file:

- [losertree.h](#)

### 5.137 `__gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >` Class Template Reference

Unstable unguarded [\\_LoserTree](#) variant storing pointers.

Inheritance diagram for `__gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >`:



### Public Member Functions

- `_LoserTreePointerUnguarded` (unsigned int \_\_k, const \_Tp &\_\_sentinel, \_Compare \_\_comp=`std::less< _Tp >()`)
- void `__delete_min_insert` (const \_Tp &\_\_key, bool \_\_sup)
- int `__get_min_source` ()
- void `__init` ()
- unsigned int `__init_winner` (unsigned int \_\_root)
- void `__insert_start` (const \_Tp &\_\_key, int \_\_source, bool)

### Protected Attributes

- `_Compare _M_comp`
- `unsigned int _M_ik`
- `unsigned int _M_k`
- `_Loser * _M_losers`
- `unsigned int _M_offset`

#### 5.137.1 Detailed Description

`template<typename _Tp, typename _Compare> class __gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >`

Unstable unguarded [\\_LoserTree](#) variant storing pointers. Stable variant is above.

Definition at line 956 of file `losertree.h`.

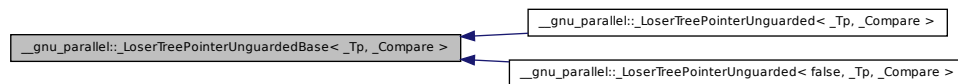
The documentation for this class was generated from the following file:

- [losertree.h](#)

### 5.138 `__gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare >` Class Template Reference

Unguarded loser tree, keeping only pointers to the elements in the tree structure.

Inheritance diagram for `__gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare >`:



### Public Member Functions

- `_LoserTreePointerUnguardedBase` (`unsigned int __k, const _Tp &__sentinel, _Compare __comp=std::less< _Tp >()`)
- `int __get_min_source ()`
- `void __insert_start` (`const _Tp &__key, int __source, bool`)

## 5.139 `__gnu_parallel::_LoserTreeTraits<_Tp>` Struct Template Reference 251

### Protected Attributes

- `_Compare` `_M_comp`
- `unsigned int` `_M_ik`
- `unsigned int` `_M_k`
- `_Loser *` `_M_losers`
- `unsigned int` `_M_offset`

#### 5.138.1 Detailed Description

`template<typename _Tp, typename _Compare> class __gnu_parallel::_LoserTreePointerUnguardedBase<_Tp, _Compare>`

Unguarded loser tree, keeping only pointers to the elements in the tree structure. No guarding is done, therefore not a single input sequence must run empty. This is a very fast variant.

Definition at line 808 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

## 5.139 `__gnu_parallel::_LoserTreeTraits<_Tp>` Struct Template Reference

Traits for determining whether the loser tree should use pointers or copies.

### Static Public Attributes

- `static const bool` `_M_use_pointer`

#### 5.139.1 Detailed Description

`template<typename _Tp> struct __gnu_parallel::_LoserTreeTraits<_Tp>`

Traits for determining whether the loser tree should use pointers or copies. The field `"_M_use_pointer"` is used to determine whether to use pointers in the loser trees or whether to copy the values into the loser tree.

The default behavior is to use pointers if the data type is 4 times as big as the pointer to it.

Specialize for your data type to customize the behavior.

Example:

```
template<> struct _LoserTreeTraits<int> { static const bool _M_use_pointer = false;
};

template<> struct _LoserTreeTraits<heavyweight_type> { static const bool _M-
use_pointer = true; };
```

#### Parameters

`_Tp` type to give the loser tree traits for.

Definition at line 727 of file `multiway_merge.h`.

#### 5.139.2 Member Data Documentation

**5.139.2.1** `template<typename _Tp > const bool __gnu_parallel::_LoserTreeTraits< _Tp >::_M_use_pointer`  
`[static]`

True iff to use pointers instead of values in loser trees.

The default behavior is to use pointers if the data type is four times as big as the pointer to it.

Definition at line 735 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

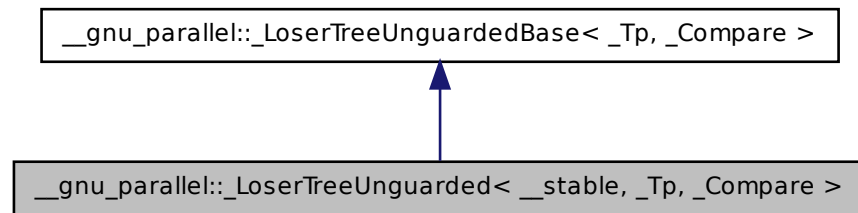
- [multiway\\_merge.h](#)

#### **5.140** `__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >` **Class Template Reference**

Stable implementation of unguarded [\\_LoserTree](#).

Inheritance diagram for `__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _`

Compare >:



### Public Member Functions

- **`_LoserTreeUnguarded`** (unsigned int \_\_k, const \_Tp \_\_sentinel, \_Compare \_\_comp=[std::less](#)< \_Tp >())
- void **`__delete_min_insert`** (\_Tp \_\_key, bool)
- int **`__get_min_source`** ()
- void **`__init`** ()
- unsigned int **`__init_winner`** (unsigned int \_\_root)
- void **`__insert_start`** (const \_Tp &\_\_key, int \_\_source, bool)

### Protected Attributes

- `_Compare _M_comp`
- unsigned int `_M_ik`
- unsigned int `_M_k`
- `_Loser * _M_losers`
- unsigned int `_M_offset`

#### 5.140.1 Detailed Description

`template<bool __stable, typename _Tp, typename _Compare> class __gnu_parallel::LoserTreeUnguarded< __stable, _Tp, _Compare >`

Stable implementation of unguarded [\\_LoserTree](#). Unstable variant is selected below with partial specialization.



### 5.141 `__gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >` Class Template Reference 1254

---

Definition at line 629 of file losertree.h.

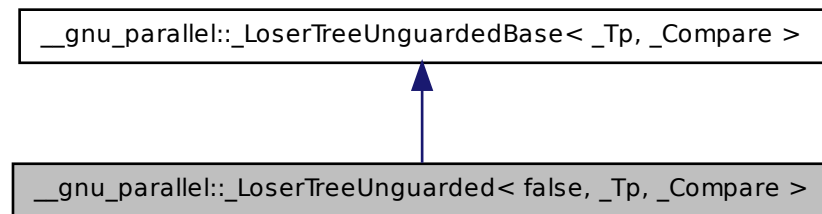
The documentation for this class was generated from the following file:

- [losertree.h](#)

### 5.141 `__gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >` Class Template Reference

Non-Stable implementation of unguarded [\\_LoserTree](#).

Inheritance diagram for `__gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >`:



#### Public Member Functions

- `_LoserTreeUnguarded` (unsigned int \_\_k, const \_Tp \_\_sentinel, \_Compare \_\_comp=[std::less](#)< \_Tp >())
- void `__delete_min_insert` (\_Tp \_\_key, bool)
- int `__get_min_source` ()
- void `__init` ()
- unsigned int `__init_winner` (unsigned int \_\_root)
- void `__insert_start` (const \_Tp &\_\_key, int \_\_source, bool)

#### Protected Attributes

- `_Compare _M_comp`
- unsigned int `_M_ik`

## 5.142 `__gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare >` Class Template Reference 1255

---

- unsigned int `_M_k`
- `_Loser * _M_losers`
- unsigned int `_M_offset`

### 5.141.1 Detailed Description

`template<typename _Tp, typename _Compare> class __gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >`

Non-Stable implementation of unguarded [\\_LoserTree](#). Stable implementation is above.

Definition at line 716 of file `losertree.h`.

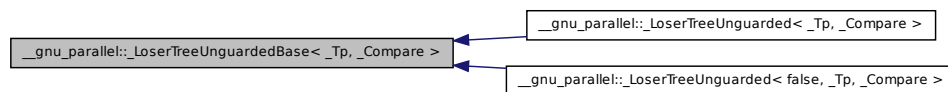
The documentation for this class was generated from the following file:

- [losertree.h](#)

## 5.142 `__gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare >` Class Template Reference

Base class for unguarded [\\_LoserTree](#) implementation.

Inheritance diagram for `__gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare >`:



### Public Member Functions

- `_LoserTreeUnguardedBase` (unsigned int `__k`, const `_Tp` `__sentinel`, `_Compare` `__comp=std::less< _Tp >()`)
- `int __get_min_source ()`
- `void __insert_start` (const `_Tp` &`__key`, int `__source`, bool)

### Protected Attributes

- `_Compare` `_M_comp`
- unsigned int `_M_ik`

### 5.143 `__gnu_parallel::_Multiplies<_Tp1, _Tp2, _Result>` Struct Template Reference 1256

---

- unsigned int `_M_k`
- `_Loser * _M_losers`
- unsigned int `_M_offset`

#### 5.142.1 Detailed Description

**template<typename `_Tp`, typename `_Compare`> class `__gnu_parallel::_LoserTreeUnguardedBase<_Tp, _Compare>`**

Base class for ungarded [\\_LoserTree](#) implementation. The whole element is copied into the tree structure.

No guarding is done, therefore not a single input sequence must run empty. Unused `__sequence` heads are marked with a sentinel which is `>` all elements that are to be merged.

This is a very fast variant.

Definition at line 566 of file `losertree.h`.

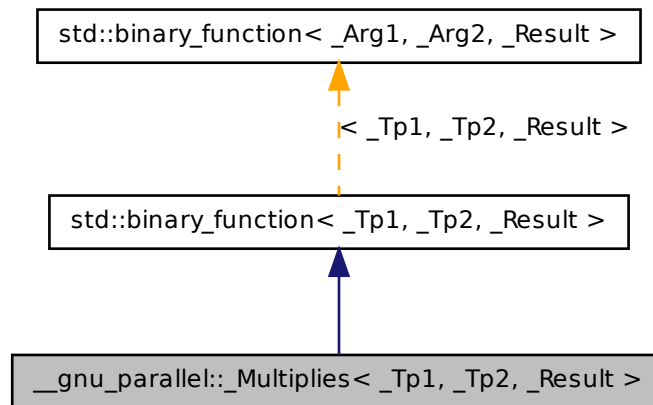
The documentation for this class was generated from the following file:

- [losertree.h](#)

### 5.143 `__gnu_parallel::_Multiplies<_Tp1, _Tp2, _Result>` Struct Template Reference

Similar to [std::multiplies](#), but allows two different types.

Inheritance diagram for `__gnu_parallel::_Multiplies< _Tp1, _Tp2, _Result >`:



## Public Types

- typedef `_Tp1` `first_argument_type`
- typedef `_Result` `result_type`
- typedef `_Tp2` `second_argument_type`

## Public Member Functions

- `_Result operator()` (`const _Tp1 &__x, const _Tp2 &__y`) `const`

### 5.143.1 Detailed Description

`template<typename _Tp1, typename _Tp2, typename _Result = __typeof__ - (*static_cast<_Tp1*>(0) * *static_cast<_Tp2*>(0))> struct __gnu_parallel::_Multiplies< _Tp1, _Tp2, _Result >`

Similar to `std::multiplies`, but allows two different types.

Definition at line 288 of file `parallel/base.h`.

### 5.143.2 Member Typedef Documentation

#### 5.143.2.1 `typedef _Tp1 std::binary_function< _Tp1 , _Tp2 , _Result >::first_argument_type [inherited]`

`first_argument_type` is the type of the first argument

Definition at line 118 of file `stl_function.h`.

#### 5.143.2.2 `typedef _Result std::binary_function< _Tp1 , _Tp2 , _Result >::result_type [inherited]`

`result_type` is the return type

Definition at line 124 of file `stl_function.h`.

#### 5.143.2.3 `typedef _Tp2 std::binary_function< _Tp1 , _Tp2 , _Result >::second_argument_type [inherited]`

`second_argument_type` is the type of the second argument

Definition at line 121 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [parallel/base.h](#)

## 5.144 `__gnu_parallel::_Nothing` Struct Reference

Functor doing nothing.

### Public Member Functions

- `template<typename _It >  
void operator\(\) (_It)`

#### 5.144.1 Detailed Description

Functor doing nothing. For some `__reduction` tasks (this is not a function object, but is passed as `__selector` `__dummy` parameter).

## **5.145 \_\_gnu\_parallel::\_Piece< \_DifferenceTp > Struct Template Referencd259**

Definition at line 288 of file for\_each\_selectors.h.

### **5.144.2 Member Function Documentation**

#### **5.144.2.1 template<typename \_It > void \_\_gnu\_parallel::\_Nothing::operator() ( \_It ) [inline]**

Functor execution.

#### **Parameters**

\_\_i iterator referencing object.

Definition at line 294 of file for\_each\_selectors.h.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## **5.145 \_\_gnu\_parallel::\_Piece< \_DifferenceTp > Struct Template Reference**

Subsequence description.

### **Public Types**

- typedef \_DifferenceTp **\_DifferenceType**

### **Public Attributes**

- \_DifferenceType [\\_M\\_begin](#)
- \_DifferenceType [\\_M\\_end](#)

### **5.145.1 Detailed Description**

**template<typename \_DifferenceTp> struct \_\_gnu\_parallel::\_Piece< \_-  
DifferenceTp >**

Subsequence description.

Definition at line 46 of file multiway\_mergesort.h.

**5.145.2 Member Data Documentation**

**5.145.2.1 template<typename \_DifferenceTp > \_DifferenceType  
\_\_gnu\_parallel::\_Piece< \_DifferenceTp >::\_M\_begin**

Begin of subsequence.

Definition at line 51 of file multiway\_mergesort.h.

**5.145.2.2 template<typename \_DifferenceTp > \_DifferenceType  
\_\_gnu\_parallel::\_Piece< \_DifferenceTp >::\_M\_end**

End of subsequence.

Definition at line 54 of file multiway\_mergesort.h.

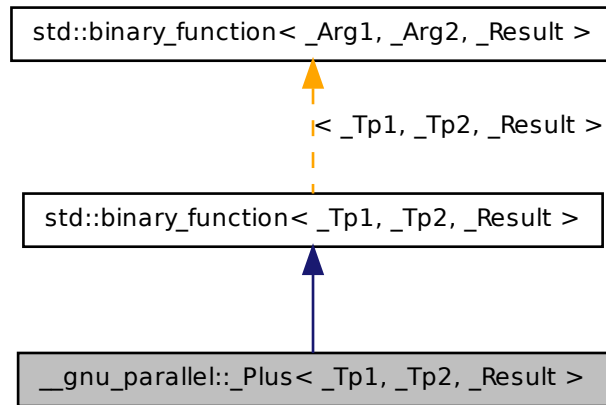
The documentation for this struct was generated from the following file:

- [multiway\\_mergesort.h](#)

**5.146 \_\_gnu\_parallel::\_Plus< \_Tp1, \_Tp2, \_Result > Struct Template Reference**

Similar to [std::plus](#), but allows two different types.

Inheritance diagram for `__gnu_parallel::_Plus< _Tp1, _Tp2, _Result >`:



### Public Types

- typedef `_Tp1` `first_argument_type`
- typedef `_Result` `result_type`
- typedef `_Tp2` `second_argument_type`

### Public Member Functions

- `_Result operator()` (`const _Tp1 &__x, const _Tp2 &__y`) `const`

#### 5.146.1 Detailed Description

`template<typename _Tp1, typename _Tp2, typename _Result = __typeof__ - (*static_cast<_Tp1*>(0) + *static_cast<_Tp2*>(0))> struct __gnu_parallel::_Plus< _Tp1, _Tp2, _Result >`

Similar to `std::plus`, but allows two different types.

Definition at line 272 of file `parallel/base.h`.



### 5.146.2 Member Typedef Documentation

5.146.2.1 `typedef _Tp1 std::binary_function<_Tp1, _Tp2, _Result>::first_argument_type [inherited]`

`first_argument_type` is the type of the first argument

Definition at line 118 of file `stl_function.h`.

5.146.2.2 `typedef _Result std::binary_function<_Tp1, _Tp2, _Result>::result_type [inherited]`

`result_type` is the return type

Definition at line 124 of file `stl_function.h`.

5.146.2.3 `typedef _Tp2 std::binary_function<_Tp1, _Tp2, _Result>::second_argument_type [inherited]`

`second_argument_type` is the type of the second argument

Definition at line 121 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [parallel/base.h](#)

## 5.147 `__gnu_parallel::_PMWMSSortingData<_RAIter>` Struct Template Reference

Data accessed by all threads.

### Public Types

- `typedef _TraitsType::difference_type` **DifferenceType**
- `typedef std::iterator_traits<_RAIter>` **\_TraitsType**
- `typedef _TraitsType::value_type` **ValueType**

## Public Attributes

- [\\_ThreadIndex](#) [\\_M\\_num\\_threads](#)
- [\\_DifferenceType](#) \* [\\_M\\_offsets](#)
- [std::vector](#)< [\\_Piece](#)< [\\_DifferenceType](#) > > \* [\\_M\\_pieces](#)
- [\\_ValueType](#) \* [\\_M\\_samples](#)
- [\\_RAIter](#) [\\_M\\_source](#)
- [\\_DifferenceType](#) \* [\\_M\\_starts](#)
- [\\_ValueType](#) \*\* [\\_M\\_temporary](#)

### 5.147.1 Detailed Description

**template<typename \_RAIter> struct `__gnu_parallel::PMWMSortingData<_RAIter>`**

Data accessed by all threads. PMWMS = parallel multiway mergesort

Definition at line 61 of file `multiway_mergesort.h`.

### 5.147.2 Member Data Documentation

**5.147.2.1 `template<typename _RAIter> _ThreadIndex  
__gnu_parallel::PMWMSortingData<_RAIter>::  
>::_M_num_threads`**

Number of threads involved.

Definition at line 68 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

**5.147.2.2 `template<typename _RAIter> _DifferenceType*  
__gnu_parallel::PMWMSortingData<_RAIter>::  
>::_M_offsets`**

Offsets to add to the found positions.

Definition at line 83 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::parallel_sort_mwms()`.

**5.147.2.3 template<typename \_RAIter> std::vector<\_Piece<\_DifferenceType> >\* \_\_gnu\_parallel::PMWMSSortingData<\_RAIter>::\_M\_pieces**

Pieces of data to merge [thread][\_\_sequence].

Definition at line 86 of file multiway\_mergesort.h.

Referenced by \_\_gnu\_parallel::parallel\_sort\_mwms(), and \_\_gnu\_parallel::parallel\_sort\_mwms\_pu().

**5.147.2.4 template<typename \_RAIter> \_ValueType\* \_\_gnu\_parallel::PMWMSSortingData<\_RAIter>::\_M\_samples**

Samples.

Definition at line 80 of file multiway\_mergesort.h.

Referenced by \_\_gnu\_parallel::\_\_determine\_samples(), and \_\_gnu\_parallel::parallel\_sort\_mwms().

**5.147.2.5 template<typename \_RAIter> \_RAIter \_\_gnu\_parallel::PMWMSSortingData<\_RAIter>::\_M\_source**

Input \_\_begin.

Definition at line 71 of file multiway\_mergesort.h.

Referenced by \_\_gnu\_parallel::\_\_determine\_samples(), \_\_gnu\_parallel::parallel\_sort\_mwms(), and \_\_gnu\_parallel::parallel\_sort\_mwms\_pu().

**5.147.2.6 template<typename \_RAIter> \_DifferenceType\* \_\_gnu\_parallel::PMWMSSortingData<\_RAIter>::\_M\_starts**

Start indices, per thread.

Definition at line 74 of file multiway\_mergesort.h.

Referenced by \_\_gnu\_parallel::\_\_determine\_samples(), \_\_gnu\_parallel::parallel\_sort\_mwms(), and \_\_gnu\_parallel::parallel\_sort\_mwms\_pu().

5.147.2.7 `template<typename _RAIter> _ValueType**  
__gnu_parallel::_PMWMSortingData< _RAIter >::_M_temporary`

Storage in which to sort.

Definition at line 77 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

The documentation for this struct was generated from the following file:

- [multiway\\_mergesort.h](#)

## 5.148 `__gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >` **Class Template Reference**

Sequence that conceptually consists of multiple copies of the same element. The copies are not stored explicitly, of course.

### Public Types

- `typedef _DifferenceTp _DifferenceType`
- `typedef \_PseudoSequenceIterator< _Tp, uint64_t > iterator`

### Public Member Functions

- [\\_PseudoSequence](#) (const \_Tp &\_\_val, \_DifferenceType \_\_count)
- [iterator begin](#) () const
- [iterator end](#) () const

#### 5.148.1 Detailed Description

`template<typename _Tp, typename _DifferenceTp> class __gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >`

Sequence that conceptually consists of multiple copies of the same element. The copies are not stored explicitly, of course.

### Parameters

- `_Tp` Sequence `_M_value` type.
- `_DifferenceTp` Sequence difference type.

Definition at line 359 of file `parallel/base.h`.

### 5.148.2 Constructor & Destructor Documentation

5.148.2.1 `template<typename _Tp, typename _DifferenceTp>  
__gnu_parallel::_PseudoSequence<_Tp, _DifferenceTp  
>::_PseudoSequence ( const _Tp & __val, _DifferenceType __count  
) [inline]`

Constructor.

#### Parameters

`__M_val` Element of the sequence.  
`__count` Number of (virtual) copies.

Definition at line 371 of file `parallel/base.h`.

### 5.148.3 Member Function Documentation

5.148.3.1 `template<typename _Tp, typename _DifferenceTp> iterator  
__gnu_parallel::_PseudoSequence<_Tp, _DifferenceTp>::begin ( ) const [inline]`

Begin iterator.

Definition at line 376 of file `parallel/base.h`.

5.148.3.2 `template<typename _Tp, typename _DifferenceTp> iterator  
__gnu_parallel::_PseudoSequence<_Tp, _DifferenceTp>::end ( )  
const [inline]`

End iterator.

Definition at line 381 of file `parallel/base.h`.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

## 5.149 `__gnu_parallel::_PseudoSequenceIterator<_Tp, _DifferenceTp>` Class Template Reference 1267

---

### 5.149 `__gnu_parallel::_PseudoSequenceIterator<_Tp, _DifferenceTp>` Class Template Reference

`_Iterator` associated with `__gnu_parallel::_PseudoSequence`. If features the usual random-access iterator functionality.

#### Public Types

- typedef `_DifferenceTp` `_DifferenceType`

#### Public Member Functions

- `_PseudoSequenceIterator` (const `_Tp` &\_\_val, `_DifferenceType` \_\_pos)
- bool `operator!=` (const `_PseudoSequenceIterator` &\_\_i2)
- const `_Tp` & `operator*` () const
- `_PseudoSequenceIterator` & `operator++` ()
- `_PseudoSequenceIterator` `operator++` (int)
- `_DifferenceType` `operator-` (const `_PseudoSequenceIterator` &\_\_i2)
- bool `operator==` (const `_PseudoSequenceIterator` &\_\_i2)
- const `_Tp` & `operator[]` (`_DifferenceType`) const

#### 5.149.1 Detailed Description

`template<typename _Tp, typename _DifferenceTp> class __gnu_parallel::_PseudoSequenceIterator<_Tp, _DifferenceTp>`

`_Iterator` associated with `__gnu_parallel::_PseudoSequence`. If features the usual random-access iterator functionality.

#### Parameters

- `_Tp` Sequence `_M_value` type.
- `_DifferenceTp` Sequence difference type.

Definition at line 306 of file `parallel/base.h`.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

### 5.150 `__gnu_parallel::_QSBThreadLocal<_RAIter>` Struct Template Reference

Information local to one thread in the parallel quicksort run.

## Public Types

- typedef `_TraitsType::difference_type` `_DifferenceType`
- typedef `std::pair<_RAIter, _RAIter>` `_Piece`
- typedef `std::iterator_traits<_RAIter>` `_TraitsType`

## Public Member Functions

- `_QSBThreadLocal` (int `__queue_size`)

## Public Attributes

- volatile `_DifferenceType` \* `_M_elements_leftover`
- `_Piece` `_M_global`
- `_Piece` `_M_initial`
- `_RestrictedBoundedConcurrentQueue<_Piece>` `_M_leftover_parts`
- `_ThreadIndex` `_M_num_threads`

### 5.150.1 Detailed Description

`template<typename _RAIter> struct __gnu_parallel::__QSBThreadLocal< _RAIter >`

Information local to one thread in the parallel quicksort run.

Definition at line 62 of file `balanced_quicksort.h`.

### 5.150.2 Member Typedef Documentation

**5.150.2.1** `template<typename _RAIter> typedef std::pair<_RAIter, _RAIter> __gnu_parallel::__QSBThreadLocal<_RAIter>::_Piece`

Continuous part of the sequence, described by an iterator pair.

Definition at line 69 of file `balanced_quicksort.h`.

### 5.150.3 Constructor & Destructor Documentation

**5.150.3.1** `template<typename _RAIter> __gnu_parallel::__QSBThreadLocal<_RAIter>::_QSBThreadLocal ( int __queue_size ) [inline]`

Constructor.

### Parameters

*\_\_queue\_size* size of the work-stealing queue.

Definition at line 88 of file `balanced_quicksort.h`.

### 5.150.4 Member Data Documentation

**5.150.4.1** `template<typename _RAIter> volatile _DifferenceType*  
__gnu_parallel::__QSBThreadLocal<_RAIter  
>::__M_elements_leftover`

Pointer to a counter of elements left over to sort.

Definition at line 81 of file `balanced_quicksort.h`.

Referenced by `__gnu_parallel::__qsb_conquer()`, and `__gnu_parallel::__qsb_local_sort_with_helping()`.

**5.150.4.2** `template<typename _RAIter> _Piece __gnu_parallel::_  
QSBThreadLocal<_RAIter>::__M_global`

The complete sequence to sort.

Definition at line 84 of file `balanced_quicksort.h`.

**5.150.4.3** `template<typename _RAIter> _Piece __gnu_parallel::_  
QSBThreadLocal<_RAIter>::__M_initial`

Initial piece to work on.

Definition at line 72 of file `balanced_quicksort.h`.

Referenced by `__gnu_parallel::__qsb_conquer()`, and `__gnu_parallel::__qsb_local_sort_with_helping()`.



**5.150.4.4** `template<typename _RAIter> _-`  
`RestrictedBoundedConcurrentQueue<_Piece>`  
`__gnu_parallel::__QSBThreadLocal<_RAIter>::__M_leftover_parts`

Work-stealing queue.

Definition at line 75 of file `balanced_quicksort.h`.

Referenced by `__gnu_parallel::__qsb_local_sort_with_helping()`.

**5.150.4.5** `template<typename _RAIter> _ThreadIndex`  
`__gnu_parallel::__QSBThreadLocal<_RAIter>::__M_num_threads`

Number of threads involved in this algorithm.

Definition at line 78 of file `balanced_quicksort.h`.

Referenced by `__gnu_parallel::__qsb_local_sort_with_helping()`.

The documentation for this struct was generated from the following file:

- [balanced\\_quicksort.h](#)

## 5.151 `__gnu_parallel::__RandomNumber` Class Reference

Random number generator, based on the Mersenne twister.

### Public Member Functions

- [\\_RandomNumber](#) ()
- [\\_RandomNumber](#) (uint32\_t \_\_seed, uint64\_t \_\_M-supremum=0x100000000ULL)
- unsigned long [\\_\\_genrand\\_bits](#) (int \_\_bits)
- uint32\_t [operator\(\)](#) ()
- uint32\_t [operator\(\)](#) (uint64\_t local\_supremum)

### 5.151.1 Detailed Description

Random number generator, based on the Mersenne twister.

Definition at line 42 of file `random_number.h`.

### 5.151.2 Constructor & Destructor Documentation

#### 5.151.2.1 \_\_gnu\_parallel::\_RandomNumber::\_RandomNumber ( ) [inline]

Default constructor. Seed with 0.

Definition at line 74 of file random\_number.h.

#### 5.151.2.2 \_\_gnu\_parallel::\_RandomNumber::\_RandomNumber ( uint32\_t \_\_seed, uint64\_t \_M\_supremum = 0x100000000ULL ) [inline]

Constructor.

##### Parameters

*\_\_seed* Random \_\_seed.

*\_M\_supremum* Generate integer random numbers in the interval [0, *\_M\_supremum*).

Definition at line 85 of file random\_number.h.

### 5.151.3 Member Function Documentation

#### 5.151.3.1 unsigned long \_\_gnu\_parallel::\_RandomNumber::\_genrand\_bits ( int *\_\_bits* ) [inline]

Generate a number of random bits, run-time parameter.

##### Parameters

*bits* Number of bits to generate.

Definition at line 109 of file random\_number.h.

#### 5.151.3.2 uint32\_t \_\_gnu\_parallel::\_RandomNumber::operator() ( ) [inline]

## 5.152 `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>` Class Template Reference 1272

---

Generate unsigned random 32-bit integer.

Definition at line 94 of file `random_number.h`.

### 5.151.3.3 `uint32_t __gnu_parallel::_RandomNumber::operator() ( uint64_t local_supremum ) [inline]`

Generate unsigned random 32-bit integer in the interval `[0,local_supremum)`.

Definition at line 100 of file `random_number.h`.

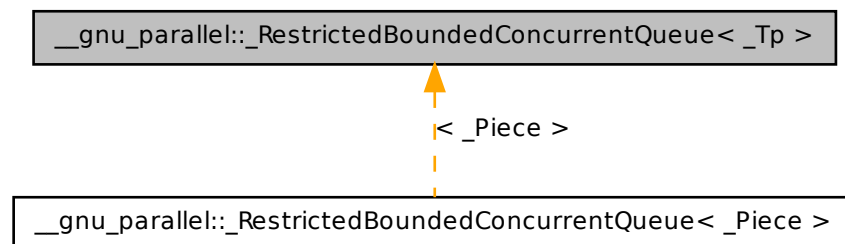
The documentation for this class was generated from the following file:

- [random\\_number.h](#)

## 5.152 `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>` Class Template Reference

Double-ended queue of bounded size, allowing lock-free atomic access. `push_front()` and `pop_front()` must not be called concurrently to each other, while `pop_back()` can be called concurrently at all times. `empty()`, `size()`, and `top()` are intentionally not provided. Calling them would not make sense in a concurrent setting.

Inheritance diagram for `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>`:



## Public Member Functions

- [\\_RestrictedBoundedConcurrentQueue](#) ([\\_SequenceIndex](#) \_\_max\_size)
- [~\\_RestrictedBoundedConcurrentQueue](#) ()
- bool [pop\\_back](#) (\_Tp &\_\_t)
- bool [pop\\_front](#) (\_Tp &\_\_t)
- void [push\\_front](#) (const \_Tp &\_\_t)

### 5.152.1 Detailed Description

**template<typename `_Tp`> class `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>`**

Double-ended queue of bounded size, allowing lock-free atomic access. [push\\_front\(\)](#) and [pop\\_front\(\)](#) must not be called concurrently to each other, while [pop\\_back\(\)](#) can be called concurrently at all times. `empty()`, `size()`, and `top()` are intentionally not provided. Calling them would not make sense in a concurrent setting.

#### Parameters

`_Tp` Contained element type.

Definition at line 52 of file `queue.h`.

### 5.152.2 Constructor & Destructor Documentation

**5.152.2.1 `template<typename _Tp> __gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>::__RestrictedBoundedConcurrentQueue ( _SequenceIndex __max_size ) [inline]`**

Constructor. Not to be called concurrent, of course.

#### Parameters

`_M_max_size` Maximal number of elements to be contained.

Definition at line 68 of file `queue.h`.

**5.152.2.2 `template<typename _Tp> __gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>::~~_RestrictedBoundedConcurrentQueue ( ) [inline]`**

Destructor. Not to be called concurrent, of course.

Definition at line 77 of file queue.h.

### 5.152.3 Member Function Documentation

**5.152.3.1** `template<typename _Tp> bool __gnu_parallel::_-`  
`RestrictedBoundedConcurrentQueue< _Tp >::pop_back ( _Tp &`  
`__t ) [inline]`

Pops one element from the queue at the front end. Must not be called concurrently with [pop\\_front\(\)](#).

Definition at line 127 of file queue.h.

**5.152.3.2** `template<typename _Tp> bool __gnu_parallel::_-`  
`RestrictedBoundedConcurrentQueue< _Tp >::pop_front ( _Tp &`  
`__t ) [inline]`

Pops one element from the queue at the front end. Must not be called concurrently with [pop\\_front\(\)](#).

Definition at line 100 of file queue.h.

Referenced by `__gnu_parallel::__qsb_local_sort_with_helping()`.

**5.152.3.3** `template<typename _Tp> void __gnu_parallel::_-`  
`RestrictedBoundedConcurrentQueue< _Tp >::push_front ( const`  
`_Tp & __t ) [inline]`

Pushes one element into the queue at the front end. Must not be called concurrently with [pop\\_front\(\)](#).

Definition at line 83 of file queue.h.

Referenced by `__gnu_parallel::__qsb_local_sort_with_helping()`.

The documentation for this class was generated from the following file:

- [queue.h](#)

**5.153** `__gnu_parallel::_SamplingSorter< __stable, _RAIter, _StrictWeakOrdering >` Struct Template Reference 1275

---

### **5.153** `__gnu_parallel::_SamplingSorter< __stable, _RAIter, _StrictWeakOrdering >` Struct Template Reference

Stable sorting functor.

#### **Public Member Functions**

- `void operator() (_RAIter __first, _RAIter __last, _StrictWeakOrdering __comp)`

#### **5.153.1 Detailed Description**

`template<bool __stable, class _RAIter, class _StrictWeakOrdering> struct __gnu_parallel::_SamplingSorter< __stable, _RAIter, _StrictWeakOrdering >`

Stable sorting functor. Used to reduce code instantiation in `multiway_merge-sampling_splitting`.

Definition at line 1004 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

### **5.154** `__gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering >` Struct Template Reference

Non-`__stable` sorting functor.

#### **Public Member Functions**

- `void operator() (_RAIter __first, _RAIter __last, _StrictWeakOrdering __comp)`

#### **5.154.1 Detailed Description**

`template<class _RAIter, class _StrictWeakOrdering> struct __gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering >`

Non-`__stable` sorting functor. Used to reduce code instantiation in `multiway_merge-sampling_splitting`.

Definition at line 1017 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

## 5.155 `__gnu_parallel::_Settings` Struct Reference

class `_Settings` /// Run-time settings for the parallel mode including all tunable parameters.

### Static Public Member Functions

- static const `_Settings` & `get` () throw ()
- static void `set` (`_Settings` &) throw ()

### Public Attributes

- `_SequenceIndex` `accumulate_minimal_n`
- unsigned int `adjacent_difference_minimal_n`
- `_AlgorithmStrategy` `algorithm_strategy`
- unsigned int `cache_line_size`
- `_SequenceIndex` `count_minimal_n`
- `_SequenceIndex` `fill_minimal_n`
- `_FindAlgorithm` `find_algorithm`
- double `find_increasing_factor`
- `_SequenceIndex` `find_initial_block_size`
- `_SequenceIndex` `find_maximum_block_size`
- float `find_scale_factor`
- `_SequenceIndex` `find_sequential_search_size`
- `_SequenceIndex` `for_each_minimal_n`
- `_SequenceIndex` `generate_minimal_n`
- unsigned long long `L1_cache_size`
- unsigned long long `L2_cache_size`
- `_SequenceIndex` `max_element_minimal_n`
- `_SequenceIndex` `merge_minimal_n`
- unsigned int `merge_oversampling`
- `_SplittingAlgorithm` `merge_splitting`
- `_SequenceIndex` `min_element_minimal_n`
- `_MultiwayMergeAlgorithm` `multiway_merge_algorithm`
- int `multiway_merge_minimal_k`
- `_SequenceIndex` `multiway_merge_minimal_n`
- unsigned int `multiway_merge_oversampling`
- `_SplittingAlgorithm` `multiway_merge_splitting`

- `_SequenceIndex` `nth_element_minimal_n`
- `_SequenceIndex` `partial_sort_minimal_n`
- `_PartialSumAlgorithm` **`partial_sum_algorithm`**
- `float` `partial_sum_dilation`
- `unsigned int` `partial_sum_minimal_n`
- `double` `partition_chunk_share`
- `_SequenceIndex` `partition_chunk_size`
- `_SequenceIndex` `partition_minimal_n`
- `_SequenceIndex` `qsb_steals`
- `unsigned int` `random_shuffle_minimal_n`
- `_SequenceIndex` `replace_minimal_n`
- `_SequenceIndex` `search_minimal_n`
- `_SequenceIndex` `set_difference_minimal_n`
- `_SequenceIndex` `set_intersection_minimal_n`
- `_SequenceIndex` `set_symmetric_difference_minimal_n`
- `_SequenceIndex` `set_union_minimal_n`
- `_SortAlgorithm` **`sort_algorithm`**
- `_SequenceIndex` `sort_minimal_n`
- `unsigned int` `sort_mwms_oversampling`
- `unsigned int` `sort_qs_num_samples_preset`
- `_SequenceIndex` `sort_qsb_base_case_maximal_n`
- `_SplittingAlgorithm` **`sort_splitting`**
- `unsigned int` `TLB_size`
- `_SequenceIndex` `transform_minimal_n`
- `_SequenceIndex` `unique_copy_minimal_n`
- `_SequenceIndex` **`workstealing_chunk_size`**

### 5.155.1 Detailed Description

class `_Settings` /// Run-time settings for the parallel mode including all tunable parameters.

Definition at line 123 of file `settings.h`.

### 5.155.2 Member Function Documentation

#### 5.155.2.1 `static const _Settings& __gnu_parallel::_Settings::get ( ) throw () [static]`

Get the global settings.



Referenced by `__gnu_parallel::__find_template()`, `__gnu_parallel::__for_each_template_random_access_workstealing()`, `__gnu_parallel::__parallel_nth_element()`, `__gnu_parallel::__parallel_partial_sum()`, `__gnu_parallel::__parallel_partial_sum_linear()`, `__gnu_parallel::__parallel_partition()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort_qs_conquer()`, `__gnu_parallel::__qsb_local_sort_with_helping()`, `__gnu_parallel::multiway_merge_sampling_splitting()`, `__gnu_parallel::parallel_multiway_merge()`, `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

**5.155.2.2** `static void __gnu_parallel::_Settings::set ( _Settings & ) throw ()`  
**[static]**

Set the global settings.

### 5.155.3 Member Data Documentation

#### 5.155.3.1 `_SequenceIndex __gnu_parallel::_Settings::accumulate_minimal_n`

Minimal input size for accumulate.

Definition at line 139 of file `settings.h`.

#### 5.155.3.2 `unsigned int __gnu_parallel::_Settings::adjacent_difference_minimal_n`

Minimal input size for adjacent\_difference.

Definition at line 142 of file `settings.h`.

#### 5.155.3.3 `unsigned int __gnu_parallel::_Settings::cache_line_size`

Overestimation of cache line size. Used to avoid false /// sharing, i.e. elements of different threads are at least this /// amount apart.

Definition at line 265 of file `settings.h`.

Referenced by `__gnu_parallel::__for_each_template_random_access_workstealing()`.

**5.155.3.4 `_SequenceIndex __gnu_parallel::_Settings::count_minimal_n`**

Minimal input size for count and count\_if.

Definition at line 145 of file settings.h.

**5.155.3.5 `_SequenceIndex __gnu_parallel::_Settings::fill_minimal_n`**

Minimal input size for fill.

Definition at line 148 of file settings.h.

**5.155.3.6 `double __gnu_parallel::_Settings::find_increasing_factor`**

Block size increase factor for find.

Definition at line 151 of file settings.h.

**5.155.3.7 `_SequenceIndex __gnu_parallel::_Settings::find_initial_block_size`**

Initial block size for find.

Definition at line 154 of file settings.h.

Referenced by `__gnu_parallel::__find_template()`.

**5.155.3.8 `_SequenceIndex __gnu_parallel::_Settings::find_maximum_block_size`**

Maximal block size for find.

Definition at line 157 of file settings.h.

**5.155.3.9 `float __gnu_parallel::_Settings::find_scale_factor`**

Block size scale-down factor with respect to current position.

Definition at line 276 of file settings.h.

Referenced by `__gnu_parallel::__find_template()`.

#### 5.155.3.10 `_SequenceIndex __gnu_parallel::_Settings::find_sequential_search_size`

Start with looking for this many elements sequentially, for find.

Definition at line 160 of file settings.h.

Referenced by `__gnu_parallel::__find_template()`.

#### 5.155.3.11 `_SequenceIndex __gnu_parallel::_Settings::for_each_minimal_n`

Minimal input size for `for_each`.

Definition at line 163 of file settings.h.

#### 5.155.3.12 `_SequenceIndex __gnu_parallel::_Settings::generate_minimal_n`

Minimal input size for `generate`.

Definition at line 166 of file settings.h.

#### 5.155.3.13 `unsigned long long __gnu_parallel::_Settings::L1_cache_size`

size of the L1 cache in bytes (underestimation).

Definition at line 254 of file settings.h.

#### 5.155.3.14 `unsigned long long __gnu_parallel::_Settings::L2_cache_size`

size of the L2 cache in bytes (underestimation).

Definition at line 257 of file settings.h.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__sequential_random_shuffle()`.

**5.155.3.15** `_SequenceIndex __gnu_parallel::_Settings::max_element_minimal_n`

Minimal input size for `max_element`.

Definition at line 169 of file `settings.h`.

**5.155.3.16** `_SequenceIndex __gnu_parallel::_Settings::merge_minimal_n`

Minimal input size for `merge`.

Definition at line 172 of file `settings.h`.

**5.155.3.17** `unsigned int __gnu_parallel::_Settings::merge_oversampling`

Oversampling factor for `merge`.

Definition at line 175 of file `settings.h`.

Referenced by `__gnu_parallel::multiway_merge_sampling_splitting()`, and `__gnu_parallel::parallel_multiway_merge()`.

**5.155.3.18** `_SequenceIndex __gnu_parallel::_Settings::min_element_minimal_n`

Minimal input size for `min_element`.

Definition at line 178 of file `settings.h`.

**5.155.3.19** `int __gnu_parallel::_Settings::multiway_merge_minimal_k`

Oversampling factor for `multiway_merge`.

Definition at line 184 of file `settings.h`.

**5.155.3.20 `_SequenceIndex __gnu_parallel::_Settings::multiway_merge_minimal_n`**

Minimal input size for `multiway_merge`.

Definition at line 181 of file `settings.h`.

**5.155.3.21 `unsigned int __gnu_parallel::_Settings::multiway_merge_oversampling`**

Oversampling factor for `multiway_merge`.

Definition at line 187 of file `settings.h`.

**5.155.3.22 `_SequenceIndex __gnu_parallel::_Settings::nth_element_minimal_n`**

Minimal input size for `nth_element`.

Definition at line 190 of file `settings.h`.

Referenced by `__gnu_parallel::__parallel_nth_element()`.

**5.155.3.23 `_SequenceIndex __gnu_parallel::_Settings::partial_sort_minimal_n`**

Minimal input size for `partial_sort`.

Definition at line 203 of file `settings.h`.

**5.155.3.24 `float __gnu_parallel::_Settings::partial_sum_dilation`**

Ratio for `partial_sum`. Assume "sum and write result" to be /// this factor slower than just "sum".

Definition at line 207 of file `settings.h`.

Referenced by `__gnu_parallel::__parallel_partial_sum_linear()`.

**5.155.3.25** `unsigned int __gnu_parallel::_Settings::partial_sum_minimal_n`

Minimal input size for `partial_sum`.

Definition at line 210 of file `settings.h`.

**5.155.3.26** `double __gnu_parallel::_Settings::partition_chunk_share`

Chunk size for partition, relative to input size. If  $> 0.0$ , /// this value overrides `partition_chunk_size`.

Definition at line 197 of file `settings.h`.

Referenced by `__gnu_parallel::__parallel_partition()`.

**5.155.3.27** `_SequenceIndex __gnu_parallel::_Settings::partition_chunk_size`

Chunk size for partition.

Definition at line 193 of file `settings.h`.

Referenced by `__gnu_parallel::__parallel_partition()`.

**5.155.3.28** `_SequenceIndex __gnu_parallel::_Settings::partition_minimal_n`

Minimal input size for partition.

Definition at line 200 of file `settings.h`.

Referenced by `__gnu_parallel::__parallel_nth_element()`.

**5.155.3.29** `_SequenceIndex __gnu_parallel::_Settings::qsb_steals`

The number of stolen ranges in load-balanced quicksort.

Definition at line 270 of file `settings.h`.

**5.155.3.30 unsigned int \_\_gnu\_parallel::\_Settings::random\_shuffle\_minimal\_n**

Minimal input size for random\_shuffle.

Definition at line 213 of file settings.h.

**5.155.3.31 \_SequenceIndex \_\_gnu\_parallel::\_Settings::replace\_minimal\_n**

Minimal input size for replace and replace\_if.

Definition at line 216 of file settings.h.

**5.155.3.32 \_SequenceIndex \_\_gnu\_parallel::\_Settings::search\_minimal\_n**

Minimal input size for search and search\_n.

Definition at line 273 of file settings.h.

**5.155.3.33 \_SequenceIndex \_\_gnu\_parallel::\_Settings::set\_difference\_minimal\_n**

Minimal input size for set\_difference.

Definition at line 219 of file settings.h.

**5.155.3.34 \_SequenceIndex \_\_gnu\_parallel::\_Settings::set\_intersection\_minimal\_n**

Minimal input size for set\_intersection.

Definition at line 222 of file settings.h.

**5.155.3.35 \_SequenceIndex \_\_gnu\_parallel::\_Settings::set\_symmetric\_difference\_minimal\_n**

Minimal input size for set\_symmetric\_difference.

Definition at line 225 of file `settings.h`.

#### 5.155.3.36 `_SequenceIndex __gnu_parallel::_Settings::set_union_minimal_n`

Minimal input size for `set_union`.

Definition at line 228 of file `settings.h`.

#### 5.155.3.37 `_SequenceIndex __gnu_parallel::_Settings::sort_minimal_n`

Minimal input size for parallel sorting.

Definition at line 231 of file `settings.h`.

#### 5.155.3.38 `unsigned int __gnu_parallel::_Settings::sort_mwms_oversampling`

Oversampling factor for parallel `std::sort` (MWMS).

Definition at line 234 of file `settings.h`.

Referenced by `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

#### 5.155.3.39 `unsigned int __gnu_parallel::_Settings::sort_qs_num_samples_preset`

Such many samples to take to find a good pivot (quicksort).

Definition at line 237 of file `settings.h`.

#### 5.155.3.40 `_SequenceIndex __gnu_parallel::_Settings::sort_qsb_base_case_maximal_n`

Maximal subsequence `__length` to switch to unbalanced `__base` case. /// Applies to `std::sort` with dynamically load-balanced quicksort.

Definition at line 241 of file `settings.h`.

Referenced by `__gnu_parallel::__qsb_local_sort_with_helping()`.



**5.156 `__gnu_parallel::SplitConsistently<__exact, _RAIter, _Compare, _SortingPlacesIterator>` Struct Template Reference** 1286

---

**5.155.3.41 `unsigned int __gnu_parallel::_Settings::TLB_size`**

size of the Translation Lookaside Buffer (underestimation).

Definition at line 260 of file settings.h.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__sequential_random_shuffle()`.

**5.155.3.42 `_SequenceIndex __gnu_parallel::_Settings::transform_minimal_n`**

Minimal input size for parallel std::transform.

Definition at line 244 of file settings.h.

**5.155.3.43 `_SequenceIndex __gnu_parallel::_Settings::unique_copy_minimal_n`**

Minimal input size for unique\_copy.

Definition at line 247 of file settings.h.

The documentation for this struct was generated from the following file:

- [settings.h](#)

**5.156 `__gnu_parallel::SplitConsistently<__exact, _RAIter, _Compare, _SortingPlacesIterator>` Struct Template Reference**

Split consistently.

**5.156.1 Detailed Description**

```
template<bool __exact, typename _RAIter, typename _Compare, typename _
SortingPlacesIterator> struct __gnu_parallel::SplitConsistently< __exact, _
RAIter, _Compare, _SortingPlacesIterator >
```

Split consistently.

Definition at line 122 of file multiway\_mergesort.h.

### 5.157 `__gnu_parallel::_SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator > Struct Template Reference` 1287

---

The documentation for this struct was generated from the following file:

- [multiway\\_mergesort.h](#)

### 5.157 `__gnu_parallel::_SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator > Struct Template Reference`

Split by sampling.

#### Public Member Functions

- `void operator() (const \_ThreadIndex __iam, \_PMWMSSortingData< \_RAIter > *__sd, \_Compare &__comp, const typename std::iterator_traits< \_RAIter >::difference_type __num_samples) const`

#### 5.157.1 Detailed Description

`template<typename \_RAIter, typename \_Compare, typename \_SortingPlacesIterator> struct __gnu_parallel::_SplitConsistently< false, \_RAIter, \_Compare, \_SortingPlacesIterator >`

Split by sampling.

Definition at line 187 of file [multiway\\_mergesort.h](#).

The documentation for this struct was generated from the following file:

- [multiway\\_mergesort.h](#)

### 5.158 `__gnu_parallel::_SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator > Struct Template Reference`

Split by exact splitting.

#### Public Member Functions

- `void operator() (const \_ThreadIndex __iam, \_PMWMSSortingData< \_RAIter > *__sd, \_Compare &__comp, const typename std::iterator_traits< \_RAIter >::difference_type __num_samples) const`

### 5.158.1 Detailed Description

```
template<typename _RAIter, typename _Compare, typename _-
SortingPlacesIterator> struct __gnu_parallel::_SplitConsistently< true, _-
RAIter, _Compare, _SortingPlacesIterator >
```

Split by exact splitting.

Definition at line 128 of file `multiway_mergesort.h`.

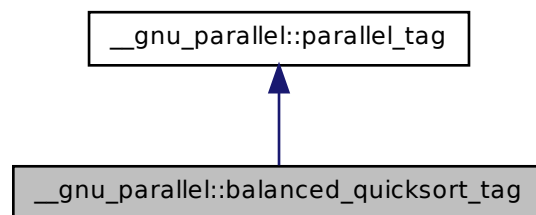
The documentation for this struct was generated from the following file:

- [multiway\\_mergesort.h](#)

## 5.159 `__gnu_parallel::balanced_quicksort_tag` Struct Reference

Forces parallel sorting using balanced quicksort at compile time.

Inheritance diagram for `__gnu_parallel::balanced_quicksort_tag`:



### Public Member Functions

- `balanced_quicksort_tag` ([\\_ThreadIndex](#) \_\_num\_threads)
- [\\_ThreadIndex](#) `__get_num_threads` ()
- `void` `set_num_threads` ([\\_ThreadIndex](#) \_\_num\_threads)

### 5.159.1 Detailed Description

Forces parallel sorting using balanced quicksort at compile time.

Definition at line 164 of file `tags.h`.

### 5.159.2 Member Function Documentation

#### 5.159.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ( )` `[inline, inherited]`

Find out desired number of threads.

##### Returns

Desired number of threads.

Definition at line 63 of file `tags.h`.

Referenced by `__gnu_parallel::__parallel_sort()`.

#### 5.159.2.2 `void __gnu_parallel::parallel_tag::set_num_threads ( _ThreadIndex __num_threads ) [inline, inherited]`

Set the desired number of threads.

##### Parameters

`__num_threads` Desired number of threads.

Definition at line 73 of file `tags.h`.

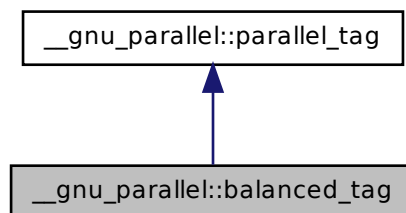
The documentation for this struct was generated from the following file:

- [tags.h](#)

### 5.160 `__gnu_parallel::balanced_tag` Struct Reference

Recommends parallel execution using dynamic load-balancing at compile time.

Inheritance diagram for \_\_gnu\_parallel::balanced\_tag:



### Public Member Functions

- [\\_ThreadIndex \\_\\_get\\_num\\_threads\(\)](#)
- void [set\\_num\\_threads\(\\_ThreadIndex \\_\\_num\\_threads\)](#)

#### 5.160.1 Detailed Description

Recommends parallel execution using dynamic load-balancing at compile time.

Definition at line 88 of file tags.h.

#### 5.160.2 Member Function Documentation

##### 5.160.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads()` [inline, inherited]

Find out desired number of threads.

### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

5.160.2.2 `void __gnu_parallel::parallel_tag::set_num_threads ( _ThreadIndex  
                   __num_threads ) [inline, inherited]`

Set the desired number of threads.

#### Parameters

*\_\_num\_threads* Desired number of threads.

Definition at line 73 of file tags.h.

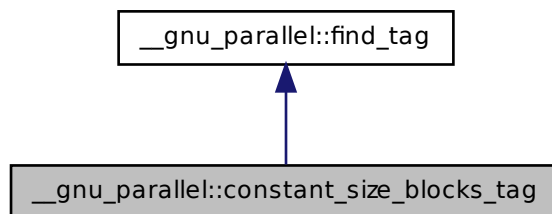
The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.161 `__gnu_parallel::constant_size_blocks_tag` Struct Reference

Selects the constant block size variant for `std::find()`.

Inheritance diagram for `__gnu_parallel::constant_size_blocks_tag`:



### 5.161.1 Detailed Description

Selects the constant block size variant for `std::find()`.

#### See also

[\\_GLIBCXX\\_FIND\\_CONSTANT\\_SIZE\\_BLOCKS](#)

Definition at line 178 of file `tags.h`.

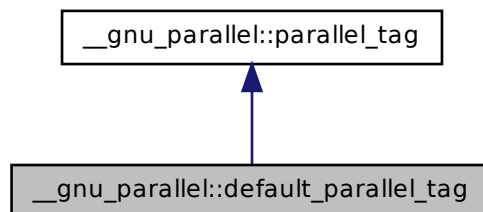
The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.162 `__gnu_parallel::default_parallel_tag` Struct Reference

Recommends parallel execution using the default parallel algorithm.

Inheritance diagram for `__gnu_parallel::default_parallel_tag`:



### Public Member Functions

- `default_parallel_tag` ([\\_ThreadIndex](#) \_\_num\_threads)
- [\\_ThreadIndex](#) `__get_num_threads` ()
- void [set\\_num\\_threads](#) ([\\_ThreadIndex](#) \_\_num\_threads)

#### 5.162.1 Detailed Description

Recommends parallel execution using the default parallel algorithm.

Definition at line 79 of file `tags.h`.

### 5.162.2 Member Function Documentation

#### 5.162.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ( )` [inline, inherited]

Find out desired number of threads.

##### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

#### 5.162.2.2 `void __gnu_parallel::parallel_tag::set_num_threads ( _ThreadIndex __num_threads )` [inline, inherited]

Set the desired number of threads.

##### Parameters

`__num_threads` Desired number of threads.

Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

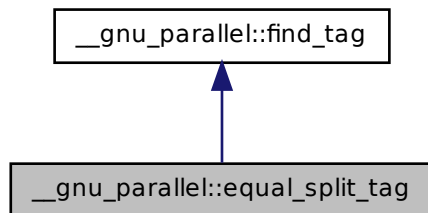
- [tags.h](#)

### 5.163 `__gnu_parallel::equal_split_tag` Struct Reference

Selects the equal splitting variant for `std::find()`.



Inheritance diagram for \_\_gnu\_parallel::equal\_split\_tag:



#### 5.163.1 Detailed Description

Selects the equal splitting variant for `std::find()`.

See also

[\\_GLIBCXX\\_FIND\\_EQUAL\\_SPLIT](#)

Definition at line 182 of file `tags.h`.

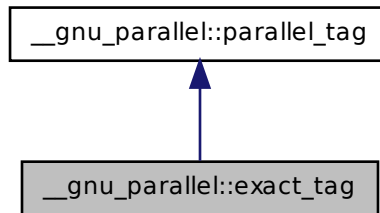
The documentation for this struct was generated from the following file:

- [tags.h](#)

#### 5.164 \_\_gnu\_parallel::exact\_tag Struct Reference

Forces parallel merging with exact splitting, at compile time.

Inheritance diagram for \_\_gnu\_parallel::exact\_tag:



### Public Member Functions

- `exact_tag` (`_ThreadIndex` \_\_num\_threads)
- `_ThreadIndex` `__get_num_threads` ()
- void `set_num_threads` (`_ThreadIndex` \_\_num\_threads)

#### 5.164.1 Detailed Description

Forces parallel merging with exact splitting, at compile time.

Definition at line 109 of file tags.h.

#### 5.164.2 Member Function Documentation

##### 5.164.2.1 `_ThreadIndex` `__gnu_parallel::parallel_tag::__get_num_threads` ( ) [inline, inherited]

Find out desired number of threads.

### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort`().

### 5.164.2.2 void \_\_gnu\_parallel::parallel\_tag::set\_num\_threads ( \_ThreadIndex \_\_num\_threads ) [inline, inherited]

Set the desired number of threads.

#### Parameters

*\_\_num\_threads* Desired number of threads.

Definition at line 73 of file tags.h.

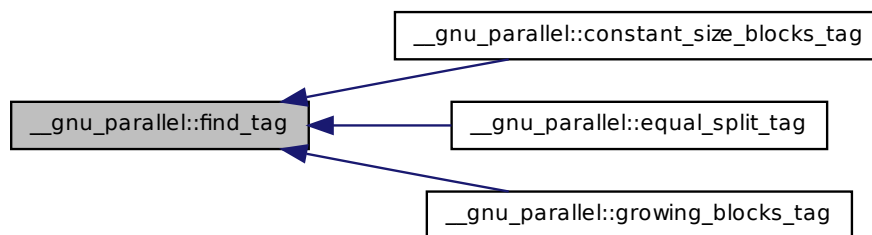
The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.165 \_\_gnu\_parallel::find\_tag Struct Reference

Base class for for std::find() variants.

Inheritance diagram for \_\_gnu\_parallel::find\_tag:



### 5.165.1 Detailed Description

Base class for for std::find() variants.

Definition at line 104 of file tags.h.

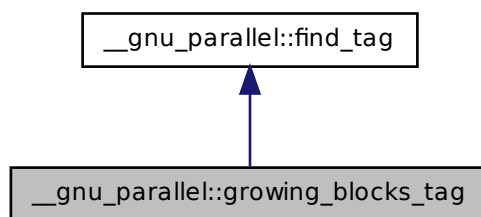
The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.166 `__gnu_parallel::growing_blocks_tag` Struct Reference

Selects the growing block size variant for `std::find()`.

Inheritance diagram for `__gnu_parallel::growing_blocks_tag`:



### 5.166.1 Detailed Description

Selects the growing block size variant for `std::find()`.

See also

[`\_GLIBCXX\_FIND\_GROWING\_BLOCKS`](#)

Definition at line 174 of file `tags.h`.

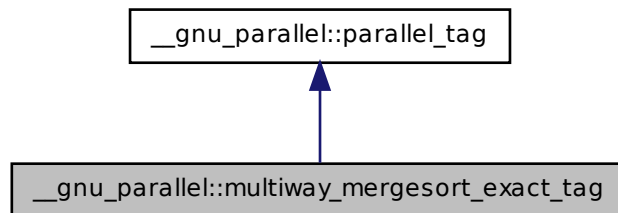
The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.167 `__gnu_parallel::multiway_mergesort_exact_tag` Struct Reference

Forces parallel sorting using multiway mergesort with exact splitting at compile time.

Inheritance diagram for `__gnu_parallel::multiway_mergesort_exact_tag`:



### Public Member Functions

- `multiway_mergesort_exact_tag` ([\\_ThreadIndex](#) \_\_num\_threads)
- [\\_ThreadIndex](#) `__get_num_threads` ()
- void `set_num_threads` ([\\_ThreadIndex](#) \_\_num\_threads)

#### 5.167.1 Detailed Description

Forces parallel sorting using multiway mergesort with exact splitting at compile time.

Definition at line 137 of file `tags.h`.

#### 5.167.2 Member Function Documentation

##### 5.167.2.1 [\\_ThreadIndex](#) `__gnu_parallel::parallel_tag::__get_num_threads` ( ) [inline, inherited]

Find out desired number of threads.

### Returns

Desired number of threads.

Definition at line 63 of file `tags.h`.

Referenced by `__gnu_parallel::__parallel_sort`().

## 5.168 `__gnu_parallel::multiway_mergesort_sampling_tag` Struct Reference 299

5.167.2.2 `void __gnu_parallel::parallel_tag::set_num_threads ( _ThreadIndex __num_threads ) [inline, inherited]`

Set the desired number of threads.

### Parameters

`__num_threads` Desired number of threads.

Definition at line 73 of file tags.h.

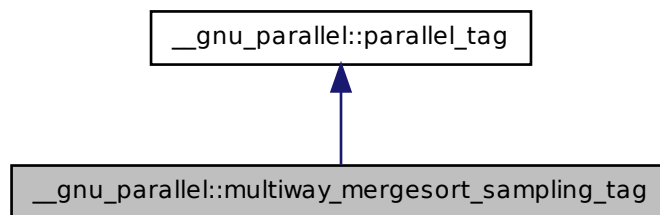
The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.168 `__gnu_parallel::multiway_mergesort_sampling_tag` Struct Reference

Forces parallel sorting using multiway mergesort with splitting by sampling at compile time.

Inheritance diagram for `__gnu_parallel::multiway_mergesort_sampling_tag`:



### Public Member Functions

- `multiway_mergesort_sampling_tag ( _ThreadIndex __num_threads )`
- `_ThreadIndex __get_num_threads ()`
- `void set_num_threads ( _ThreadIndex __num_threads )`

### 5.168.1 Detailed Description

Forces parallel sorting using multiway mergesort with splitting by sampling at compile time.

Definition at line 146 of file tags.h.

### 5.168.2 Member Function Documentation

#### 5.168.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ( )` [inline, inherited]

Find out desired number of threads.

#### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

#### 5.168.2.2 `void __gnu_parallel::parallel_tag::set_num_threads ( _ThreadIndex __num_threads )` [inline, inherited]

Set the desired number of threads.

#### Parameters

`__num_threads` Desired number of threads.

Definition at line 73 of file tags.h.

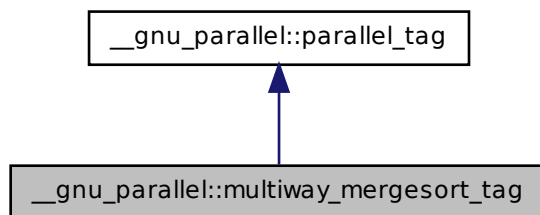
The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.169 `__gnu_parallel::multiway_mergesort_tag` Struct Reference

Forces parallel sorting using multiway mergesort at compile time.

Inheritance diagram for `__gnu_parallel::multiway_mergesort_tag`:



### Public Member Functions

- `multiway_mergesort_tag` (`_ThreadIndex` \_\_num\_threads)
- `_ThreadIndex` `__get_num_threads` ()
- void `set_num_threads` (`_ThreadIndex` \_\_num\_threads)

#### 5.169.1 Detailed Description

Forces parallel sorting using multiway mergesort at compile time.

Definition at line 128 of file tags.h.

#### 5.169.2 Member Function Documentation

##### 5.169.2.1 `_ThreadIndex` `__gnu_parallel::parallel_tag::__get_num_threads` ( ) [inline, inherited]

Find out desired number of threads.

### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort`().



5.169.2.2 `void __gnu_parallel::parallel_tag::set_num_threads ( _ThreadIndex __num_threads ) [inline, inherited]`

Set the desired number of threads.

#### Parameters

`__num_threads` Desired number of threads.

Definition at line 73 of file tags.h.

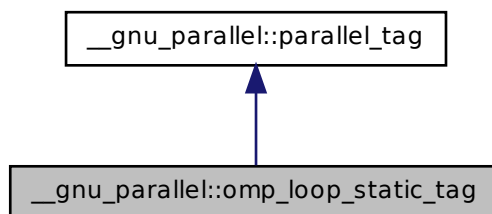
The documentation for this struct was generated from the following file:

- [tags.h](#)

### 5.170 `__gnu_parallel::omp_loop_static_tag` Struct Reference

Recommends parallel execution using OpenMP static load-balancing at compile time.

Inheritance diagram for `__gnu_parallel::omp_loop_static_tag`:



#### Public Member Functions

- `_ThreadIndex __get_num_threads ()`
- `void set_num_threads (_ThreadIndex __num_threads)`

#### 5.170.1 Detailed Description

Recommends parallel execution using OpenMP static load-balancing at compile time.

Definition at line 100 of file tags.h.

### 5.170.2 Member Function Documentation

#### 5.170.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ( )` `[inline, inherited]`

Find out desired number of threads.

##### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

#### 5.170.2.2 `void __gnu_parallel::parallel_tag::set_num_threads ( _ThreadIndex` `__num_threads ) [inline, inherited]`

Set the desired number of threads.

##### Parameters

`__num_threads` Desired number of threads.

Definition at line 73 of file tags.h.

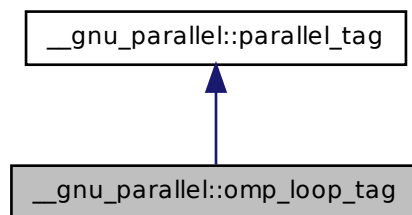
The documentation for this struct was generated from the following file:

- [tags.h](#)

### 5.171 `__gnu_parallel::omp_loop_tag` Struct Reference

Recommends parallel execution using OpenMP dynamic load-balancing at compile time.

Inheritance diagram for \_\_gnu\_parallel::omp\_loop\_tag:



### Public Member Functions

- [\\_ThreadIndex \\_\\_get\\_num\\_threads\(\)](#)
- void [set\\_num\\_threads\(\\_ThreadIndex \\_\\_num\\_threads\)](#)

#### 5.171.1 Detailed Description

Recommends parallel execution using OpenMP dynamic load-balancing at compile time.

Definition at line 96 of file tags.h.

#### 5.171.2 Member Function Documentation

##### 5.171.2.1 [\\_ThreadIndex \\_\\_gnu\\_parallel::parallel\\_tag::\\_\\_get\\_num\\_threads\(\)](#) [inline, inherited]

Find out desired number of threads.

### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

**5.171.2.2** `void __gnu_parallel::parallel_tag::set_num_threads ( _ThreadIndex  
__num_threads ) [inline, inherited]`

Set the desired number of threads.

#### Parameters

`__num_threads` Desired number of threads.

Definition at line 73 of file `tags.h`.

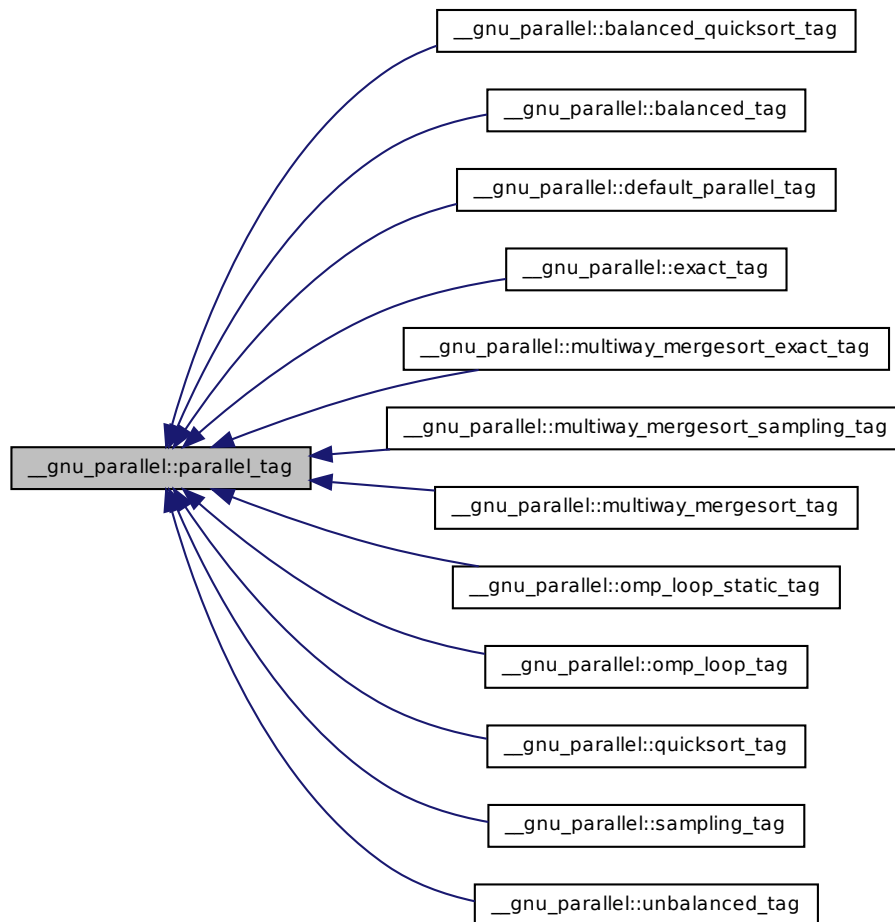
The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.172 `__gnu_parallel::parallel_tag` Struct Reference

Recommends parallel execution at compile time, optionally using a user-specified number of threads.

Inheritance diagram for \_\_gnu\_parallel::parallel\_tag:



### Public Member Functions

- [parallel\\_tag \(\)](#)
- [parallel\\_tag \(\\_ThreadIndex \\_\\_num\\_threads\)](#)
- [\\_ThreadIndex \\_\\_get\\_num\\_threads \(\)](#)
- [void set\\_num\\_threads \(\\_ThreadIndex \\_\\_num\\_threads\)](#)

### 5.172.1 Detailed Description

Recommends parallel execution at compile time, optionally using a user-specified number of threads.

Definition at line 46 of file tags.h.

### 5.172.2 Constructor & Destructor Documentation

#### 5.172.2.1 `__gnu_parallel::parallel_tag::parallel_tag ( ) [inline]`

Default constructor. Use default number of threads.

Definition at line 53 of file tags.h.

#### 5.172.2.2 `__gnu_parallel::parallel_tag::parallel_tag ( _ThreadIndex __num_threads ) [inline]`

Default constructor. Recommend number of threads to use.

#### Parameters

`__num_threads` Desired number of threads.

Definition at line 58 of file tags.h.

### 5.172.3 Member Function Documentation

#### 5.172.3.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ( ) [inline]`

Find out desired number of threads.

#### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

### 5.172.3.2 `void __gnu_parallel::parallel_tag::set_num_threads ( _ThreadIndex __num_threads ) [inline]`

Set the desired number of threads.

#### Parameters

`__num_threads` Desired number of threads.

Definition at line 73 of file `tags.h`.

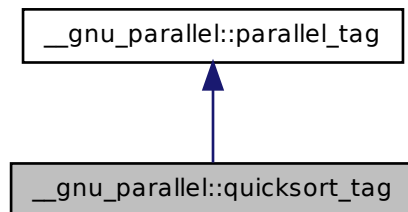
The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.173 `__gnu_parallel::quicksort_tag` Struct Reference

Forces parallel sorting using unbalanced quicksort at compile time.

Inheritance diagram for `__gnu_parallel::quicksort_tag`:



#### Public Member Functions

- `quicksort_tag ( _ThreadIndex __num_threads )`
- `_ThreadIndex __get_num_threads ()`
- `void set_num_threads ( _ThreadIndex __num_threads )`

### 5.173.1 Detailed Description

Forces parallel sorting using unbalanced quicksort at compile time.

Definition at line 155 of file `tags.h`.

### 5.173.2 Member Function Documentation

#### 5.173.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ( )` [`inline`, `inherited`]

Find out desired number of threads.

#### Returns

Desired number of threads.

Definition at line 63 of file `tags.h`.

Referenced by `__gnu_parallel::__parallel_sort()`.

#### 5.173.2.2 `void __gnu_parallel::parallel_tag::set_num_threads ( _ThreadIndex __num_threads )` [`inline`, `inherited`]

Set the desired number of threads.

#### Parameters

`__num_threads` Desired number of threads.

Definition at line 73 of file `tags.h`.

The documentation for this struct was generated from the following file:

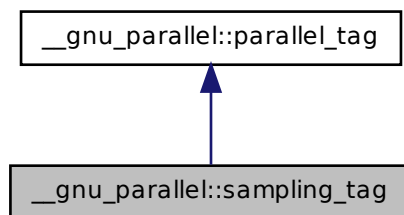
- [tags.h](#)

## 5.174 `__gnu_parallel::sampling_tag` Struct Reference

Forces parallel merging with exact splitting, at compile time.



Inheritance diagram for `__gnu_parallel::sampling_tag`:



### Public Member Functions

- `sampling_tag` (`_ThreadIndex` \_\_num\_threads)
- `_ThreadIndex` `__get_num_threads` ()
- void `set_num_threads` (`_ThreadIndex` \_\_num\_threads)

#### 5.174.1 Detailed Description

Forces parallel merging with exact splitting, at compile time.

Definition at line 118 of file tags.h.

#### 5.174.2 Member Function Documentation

##### 5.174.2.1 `_ThreadIndex` `__gnu_parallel::parallel_tag::__get_num_threads` ( ) [inline, inherited]

Find out desired number of threads.

### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort`().

### 5.174.2.2 `void __gnu_parallel::parallel_tag::set_num_threads ( _ThreadIndex __num_threads ) [inline, inherited]`

Set the desired number of threads.

#### Parameters

`__num_threads` Desired number of threads.

Definition at line 73 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.175 `__gnu_parallel::sequential_tag` Struct Reference

Forces sequential execution at compile time.

### 5.175.1 Detailed Description

Forces sequential execution at compile time.

Definition at line 42 of file `tags.h`.

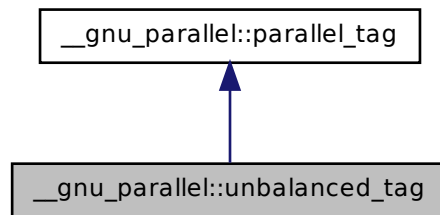
The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.176 `__gnu_parallel::unbalanced_tag` Struct Reference

Recommends parallel execution using static load-balancing at compile time.

Inheritance diagram for `__gnu_parallel::unbalanced_tag`:



### Public Member Functions

- [`\_ThreadIndex \_\_get\_num\_threads\(\)`](#)
- void [`set\_num\_threads\(\_ThreadIndex \_\_num\_threads\)`](#)

#### 5.176.1 Detailed Description

Recommends parallel execution using static load-balancing at compile time.

Definition at line 92 of file `tags.h`.

#### 5.176.2 Member Function Documentation

##### 5.176.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads()` [inline, inherited]

Find out desired number of threads.

### Returns

Desired number of threads.

Definition at line 63 of file `tags.h`.

Referenced by `__gnu_parallel::__parallel_sort()`.

### 5.176.2.2 `void __gnu_parallel::parallel_tag::set_num_threads ( _ThreadIndex __num_threads ) [inline, inherited]`

Set the desired number of threads.

#### Parameters

`__num_threads` Desired number of threads.

Definition at line 73 of file `tags.h`.

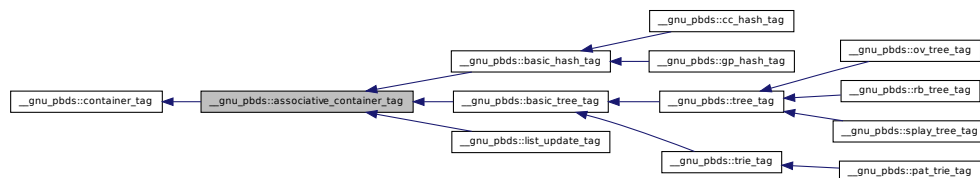
The documentation for this struct was generated from the following file:

- [tags.h](#)

## 5.177 `__gnu_pbds::associative_container_tag` Struct Reference

Basic associative-container.

Inheritance diagram for `__gnu_pbds::associative_container_tag`:



### 5.177.1 Detailed Description

Basic associative-container.

Definition at line 103 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

**5.178** `__gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_TL, Allocator >` Class Template Reference 1314

**5.178** `__gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_TL, Allocator >` Class Template Reference

An abstract basic hash-based associative container.

Inheritance diagram for `__gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_TL, Allocator >`:



### Public Types

- typedef Allocator **allocator\_type**
- typedef base\_type::const\_iterator **const\_iterator**
- typedef key\_rebind::const\_pointer **const\_key\_pointer**
- typedef key\_rebind::const\_reference **const\_key\_reference**
- typedef mapped\_rebind::const\_pointer **const\_mapped\_pointer**
- typedef mapped\_rebind::const\_reference **const\_mapped\_reference**
- typedef base\_type::const\_point\_iterator **const\_point\_iterator**
- typedef value\_rebind::const\_pointer **const\_pointer**
- typedef value\_rebind::const\_reference **const\_reference**
- typedef Tag **container\_category**
- typedef allocator\_type::difference\_type **difference\_type**
- typedef base\_type::iterator **iterator**
- typedef key\_rebind::pointer **key\_pointer**
- typedef allocator\_type::template rebind< key\_type >::other **key\_rebind**
- typedef key\_rebind::reference **key\_reference**
- typedef allocator\_type::template rebind< Key >::other::value\_type **key\_type**
- typedef mapped\_rebind::pointer **mapped\_pointer**
- typedef allocator\_type::template rebind< mapped\_type >::other **mapped\_rebind**
- typedef mapped\_rebind::reference **mapped\_reference**
- typedef Mapped **mapped\_type**
- typedef base\_type::point\_iterator **point\_iterator**
- typedef value\_rebind::pointer **pointer**
- typedef value\_rebind::reference **reference**
- typedef allocator\_type::size\_type **size\_type**
- typedef allocator\_type::template rebind< value\_type >::other **value\_rebind**
- typedef base\_type::value\_type **value\_type**

### 5.178.1 Detailed Description

**template<typename Key, typename Mapped, typename Hash\_Fn, typename Eq\_Fn, typename Resize\_Policy, bool Store\_Hash, typename Tag, typename Policy\_TL, typename Allocator> class `__gnu_pbds::basic_hash_table`< Key, Mapped, Hash\_Fn, Eq\_Fn, Resize\_Policy, Store\_Hash, Tag, Policy\_TL, Allocator >**

An abstract basic hash-based associative container.

Definition at line 144 of file `assoc_container.hpp`.

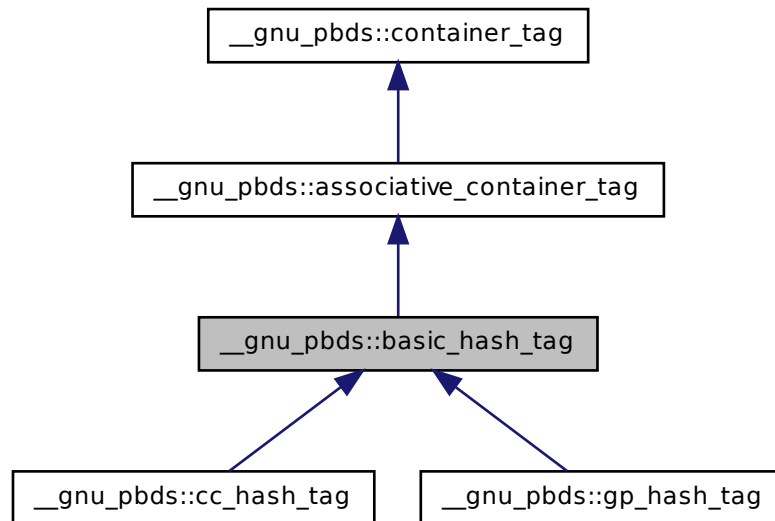
The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

## 5.179 `__gnu_pbds::basic_hash_tag` Struct Reference

Basic hash.

Inheritance diagram for `__gnu_pbds::basic_hash_tag`:



### 5.179.1 Detailed Description

Basic hash.

Definition at line 106 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.180 `__gnu_pbds::basic_tree< Key, Mapped, Tag, Node_Update, Policy_Tl, Allocator >` Class Template Reference

An abstract basic tree-like (tree, trie) associative container.

Inheritance diagram for `__gnu_pbds::basic_tree< Key, Mapped, Tag, Node_Update, Policy_Tl, Allocator >`:



### Public Types

- typedef Allocator **allocator\_type**
- typedef base\_type::const\_iterator **const\_iterator**
- typedef key\_rebind::const\_pointer **const\_key\_pointer**
- typedef key\_rebind::const\_reference **const\_key\_reference**
- typedef mapped\_rebind::const\_pointer **const\_mapped\_pointer**
- typedef mapped\_rebind::const\_reference **const\_mapped\_reference**
- typedef base\_type::const\_point\_iterator **const\_point\_iterator**
- typedef value\_rebind::const\_pointer **const\_pointer**
- typedef value\_rebind::const\_reference **const\_reference**
- typedef Tag **container\_category**
- typedef allocator\_type::difference\_type **difference\_type**
- typedef base\_type::iterator **iterator**
- typedef key\_rebind::pointer **key\_pointer**
- typedef allocator\_type::template rebind< key\_type >::other **key\_rebind**
- typedef key\_rebind::reference **key\_reference**
- typedef allocator\_type::template rebind< Key >::other::value\_type **key\_type**
- typedef mapped\_rebind::pointer **mapped\_pointer**
- typedef allocator\_type::template rebind< mapped\_type >::other **mapped\_rebind**
- typedef mapped\_rebind::reference **mapped\_reference**
- typedef Mapped **mapped\_type**

- typedef Node\_Update **node\_update**
- typedef base\_type::point\_iterator **point\_iterator**
- typedef value\_rebind::pointer **pointer**
- typedef value\_rebind::reference **reference**
- typedef allocator\_type::size\_type **size\_type**
- typedef allocator\_type::template rebind< value\_type >::other **value\_rebind**
- typedef base\_type::value\_type **value\_type**

### 5.180.1 Detailed Description

template<typename Key, typename Mapped, typename Tag, typename Node\_Update, typename Policy\_Tl, typename Allocator> class `__gnu_pbds::basic_tree`< Key, Mapped, Tag, Node\_Update, Policy\_Tl, Allocator >

An abstract basic tree-like (tree, trie) associative container.

Definition at line 477 of file `assoc_container.hpp`.

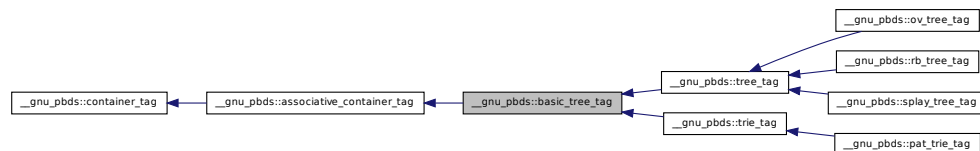
The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

## 5.181 `__gnu_pbds::basic_tree_tag` Struct Reference

Basic tree.

Inheritance diagram for `__gnu_pbds::basic_tree_tag`:



### 5.181.1 Detailed Description

Basic tree.

Definition at line 115 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

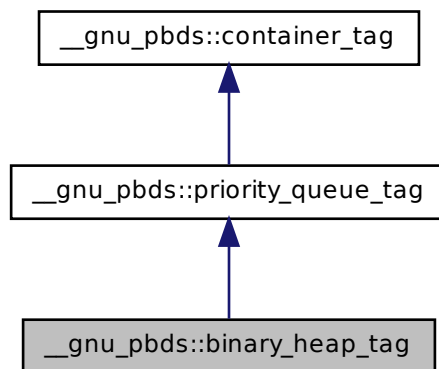
- [tag\\_and\\_trait.hpp](#)



## 5.182 \_\_gnu\_pbds::binary\_heap\_tag Struct Reference

Binary-heap (array-based).

Inheritance diagram for \_\_gnu\_pbds::binary\_heap\_tag:



### 5.182.1 Detailed Description

Binary-heap (array-based).

Definition at line 151 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

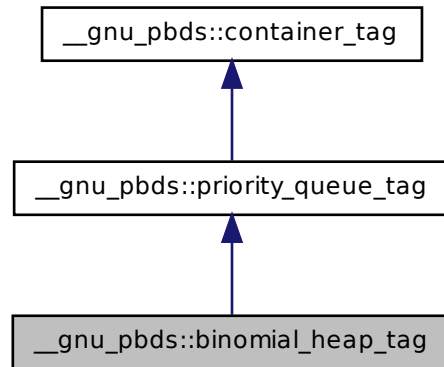
## 5.183 \_\_gnu\_pbds::binomial\_heap\_tag Struct Reference

Binomial-heap.

**5.184** `__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, Allocator >` Class Template Reference

1319

Inheritance diagram for `__gnu_pbds::binomial_heap_tag`:



**5.183.1 Detailed Description**

Binomial-heap.

Definition at line 145 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

**5.184** `__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, Allocator >` Class Template Reference

A concrete collision-chaining hash-based associative container.

Inheritance diagram for `__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, Allocator >`:



## Public Types

- typedef Allocator **allocator\_type**
- typedef Comb\_Hash\_Fn **comb\_hash\_fn**
- typedef base\_type::const\_iterator **const\_iterator**
- typedef key\_rebind::const\_pointer **const\_key\_pointer**
- typedef key\_rebind::const\_reference **const\_key\_reference**
- typedef mapped\_rebind::const\_pointer **const\_mapped\_pointer**
- typedef mapped\_rebind::const\_reference **const\_mapped\_reference**
- typedef base\_type::const\_point\_iterator **const\_point\_iterator**
- typedef value\_rebind::const\_pointer **const\_pointer**
- typedef value\_rebind::const\_reference **const\_reference**
- typedef [cc\\_hash\\_tag](#) **container\_category**
- typedef allocator\_type::difference\_type **difference\_type**
- typedef Eq\_Fn **eq\_fn**
- typedef Hash\_Fn **hash\_fn**
- typedef base\_type::iterator **iterator**
- typedef key\_rebind::pointer **key\_pointer**
- typedef allocator\_type::template rebind< key\_type >::other **key\_rebind**
- typedef key\_rebind::reference **key\_reference**
- typedef allocator\_type::template rebind< Key >::other::value\_type **key\_type**
- typedef mapped\_rebind::pointer **mapped\_pointer**
- typedef allocator\_type::template rebind< mapped\_type >::other **mapped\_rebind**
- typedef mapped\_rebind::reference **mapped\_reference**
- typedef Mapped **mapped\_type**
- typedef base\_type::point\_iterator **point\_iterator**
- typedef value\_rebind::pointer **pointer**
- typedef value\_rebind::reference **reference**
- typedef Resize\_Policy **resize\_policy**
- typedef allocator\_type::size\_type **size\_type**
- typedef allocator\_type::template rebind< value\_type >::other **value\_rebind**
- typedef base\_type::value\_type **value\_type**

## Public Member Functions

- **cc\_hash\_table** (const hash\_fn &h)
- **cc\_hash\_table** (const hash\_fn &h, const eq\_fn &e, const comb\_hash\_fn &ch)
- template<typename It >  
  **cc\_hash\_table** (It first, It last, const hash\_fn &h, const eq\_fn &e)
- **cc\_hash\_table** (const [cc\\_hash\\_table](#) &other)

- `template<typename It >`  
`cc_hash_table` (It first, It last, const hash\_fn &h, const eq\_fn &e, const comb\_hash\_fn &ch, const resize\_policy &rp)
- `template<typename It >`  
`cc_hash_table` (It first, It last, const hash\_fn &h, const eq\_fn &e, const comb\_hash\_fn &ch)
- `cc_hash_table` (const hash\_fn &h, const eq\_fn &e, const comb\_hash\_fn &ch, const resize\_policy &rp)
- `template<typename It >`  
`cc_hash_table` (It first, It last, const hash\_fn &h)
- `template<typename It >`  
`cc_hash_table` (It first, It last)
- `cc_hash_table` (const hash\_fn &h, const eq\_fn &e)
- `cc_hash_table` & **operator=** (const `cc_hash_table` &other)
- void **swap** (`cc_hash_table` &other)

#### 5.184.1 Detailed Description

`template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename Allocator = std::allocator<char>> class __gnu_pbds::cc_hash_table<Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, Allocator >`

A concrete collision-chaining hash-based associative container.

Definition at line 180 of file `assoc_container.hpp`.

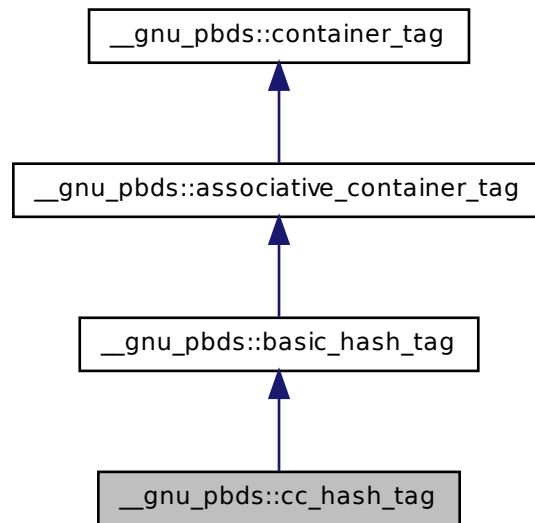
The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

## 5.185 `__gnu_pbds::cc_hash_tag` Struct Reference

Collision-chaining hash.

Inheritance diagram for `__gnu_pbds::cc_hash_tag`:



#### 5.185.1 Detailed Description

Collision-chaining hash.

Definition at line 109 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

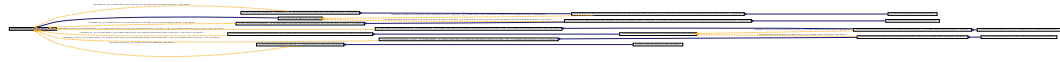
- [tag\\_and\\_trait.hpp](#)

#### 5.186 `__gnu_pbds::container_base< Key, Mapped, Tag, Policy_Tl, Allocator >` Class Template Reference

An abstract basic associative container.

Inheritance diagram for `__gnu_pbds::container_base< Key, Mapped, Tag, Policy_Tl,`

Allocator >:



## Public Types

- typedef Allocator **allocator\_type**
- typedef base\_type::const\_iterator **const\_iterator**
- typedef key\_rebind::const\_pointer **const\_key\_pointer**
- typedef key\_rebind::const\_reference **const\_key\_reference**
- typedef mapped\_rebind::const\_pointer **const\_mapped\_pointer**
- typedef mapped\_rebind::const\_reference **const\_mapped\_reference**
- typedef base\_type::const\_point\_iterator **const\_point\_iterator**
- typedef value\_rebind::const\_pointer **const\_pointer**
- typedef value\_rebind::const\_reference **const\_reference**
- typedef Tag **container\_category**
- typedef allocator\_type::difference\_type **difference\_type**
- typedef base\_type::iterator **iterator**
- typedef key\_rebind::pointer **key\_pointer**
- typedef allocator\_type::template rebind< key\_type >::other **key\_rebind**
- typedef key\_rebind::reference **key\_reference**
- typedef allocator\_type::template rebind< Key >::other::value\_type **key\_type**
- typedef mapped\_rebind::pointer **mapped\_pointer**
- typedef allocator\_type::template rebind< mapped\_type >::other **mapped\_rebind**
- typedef mapped\_rebind::reference **mapped\_reference**
- typedef Mapped **mapped\_type**
- typedef base\_type::point\_iterator **point\_iterator**
- typedef value\_rebind::pointer **pointer**
- typedef value\_rebind::reference **reference**
- typedef allocator\_type::size\_type **size\_type**
- typedef allocator\_type::template rebind< value\_type >::other **value\_rebind**
- typedef base\_type::value\_type **value\_type**

### 5.186.1 Detailed Description

`template<typename Key, typename Mapped, typename Tag, typename Policy_Tl, typename Allocator> class __gnu_pbds::container_base< Key, Mapped, Tag, Policy_Tl, Allocator >`

An abstract basic associative container.

Definition at line 77 of file `assoc_container.hpp`.

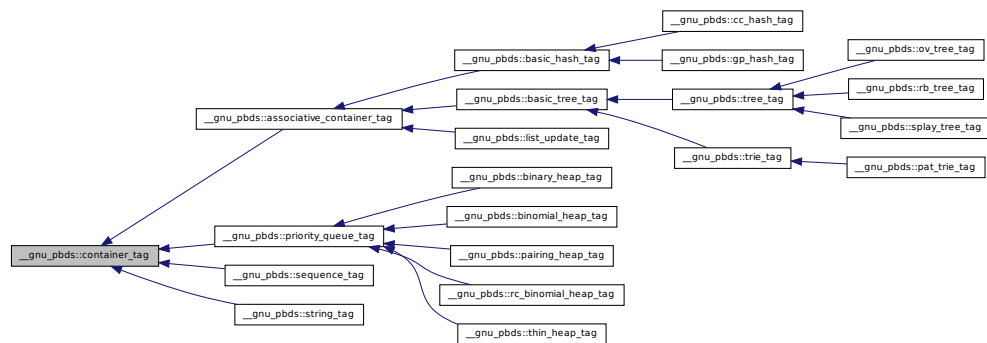
The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

## 5.187 `__gnu_pbds::container_tag` Struct Reference

Base data structure tag.

Inheritance diagram for `__gnu_pbds::container_tag`:



### 5.187.1 Detailed Description

Base data structure tag.

Definition at line 93 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.188 `__gnu_pbds::container_traits< Cntnr >` Struct Template Reference

[container\\_traits](#)

Inherits `container_traits_base< Cntnr::container_category >`.

### Public Types

- enum { **order\_preserving**, **erase\_can\_throw**, **split\_join\_can\_throw**, **reverse\_iteration** }
- typedef container\_traits\_base< container\_category > **base\_type**
- typedef Cntnr::container\_category **container\_category**
- typedef Cntnr **container\_type**
- typedef base\_type::invalidation\_guarantee **invalidation\_guarantee**

#### 5.188.1 Detailed Description

template<typename Cntnr> struct `__gnu_pbds::container_traits< Cntnr >`

[container\\_traits](#)

Definition at line 346 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 5.189 `__gnu_pbds::detail::value_type_base< Key, Mapped, Allocator, false >` Struct Template Reference

#### Public Types

- typedef mapped\_type\_allocator::const\_pointer **const\_mapped\_pointer**
- typedef mapped\_type\_allocator::const\_reference **const\_mapped\_reference**
- typedef value\_type\_allocator::const\_pointer **const\_pointer**
- typedef value\_type\_allocator::const\_reference **const\_reference**
- typedef mapped\_type\_allocator::pointer **mapped\_pointer**
- typedef mapped\_type\_allocator::reference **mapped\_reference**
- typedef mapped\_type\_allocator::value\_type **mapped\_type**
- typedef Allocator::template rebind< Mapped >::other **mapped\_type\_allocator**
- typedef value\_type\_allocator::pointer **pointer**
- typedef value\_type\_allocator::reference **reference**
- typedef value\_type\_allocator::value\_type **value\_type**
- typedef Allocator::template rebind< [std::pair](#)< const Key, Mapped > >::other **value\_type\_allocator**



#### 5.189.1 Detailed Description

**template<typename Key, typename Mapped, typename Allocator> struct `__gnu_pbds::detail::value_type_base< Key, Mapped, Allocator, false >`**

Specialization of `value_type_base` for the case where the hash value is not stored alongside each value.

Definition at line 61 of file `basic_types.hpp`.

The documentation for this struct was generated from the following file:

- [basic\\_types.hpp](#)

#### 5.190 `__gnu_pbds::detail::value_type_base< Key, Mapped, Allocator, true >` Struct Template Reference

##### Public Types

- `typedef mapped_type_allocator::const_pointer const_mapped_pointer`
- `typedef mapped_type_allocator::const_reference const_mapped_reference`
- `typedef value_type_allocator::const_pointer const_pointer`
- `typedef value_type_allocator::const_reference const_reference`
- `typedef mapped_type_allocator::pointer mapped_pointer`
- `typedef mapped_type_allocator::reference mapped_reference`
- `typedef mapped_type_allocator::value_type mapped_type`
- `typedef Allocator::template rebind< Mapped >::other mapped_type_allocator`
  
- `typedef value_type_allocator::pointer pointer`
- `typedef value_type_allocator::reference reference`
- `typedef value_type_allocator::value_type value_type`
- `typedef Allocator::template rebind< std::pair< const Key, Mapped > >::other value_type_allocator`

#### 5.190.1 Detailed Description

**template<typename Key, typename Mapped, typename Allocator> struct `__gnu_pbds::detail::value_type_base< Key, Mapped, Allocator, true >`**

Specialization of `value_type_base` for the case where the hash value is stored alongside each value.

Definition at line 88 of file `basic_types.hpp`.

The documentation for this struct was generated from the following file:

- [basic\\_types.hpp](#)

## 5.191 `__gnu_pbds::detail::value_type_base< Key, null_mapped_type, Allocator, false >` Struct Template Reference

### Public Types

- `typedef mapped_type_allocator::const_pointer` **const\_mapped\_pointer**
- `typedef mapped_type_allocator::const_reference` **const\_mapped\_reference**
- `typedef value_type_allocator::const_pointer` **const\_pointer**
- `typedef value_type_allocator::const_reference` **const\_reference**
- `typedef mapped_type_allocator::pointer` **mapped\_pointer**
- `typedef mapped_type_allocator::reference` **mapped\_reference**
- `typedef mapped_type_allocator::value_type` **mapped\_type**
- `typedef Allocator::template rebind< null\_mapped\_type >::other` **mapped\_type\_allocator**
- `typedef value_type_allocator::pointer` **pointer**
- `typedef value_type_allocator::reference` **reference**
- `typedef Key` **value\_type**
- `typedef Allocator::template rebind< value_type >::other` **value\_type\_allocator**

### Static Public Attributes

- static [null\\_mapped\\_type](#) **s\_null\_mapped**

#### 5.191.1 Detailed Description

`template<typename Key, typename Allocator> struct __gnu_pbds::detail::value_type_base< Key, null_mapped_type, Allocator, false >`

Specialization of `value_type_base` for the case where the hash value is not stored alongside each value.

Definition at line 122 of file `basic_types.hpp`.

The documentation for this struct was generated from the following file:

- [basic\\_types.hpp](#)

### 5.192 `__gnu_pbds::detail::value_type_base< Key, null_mapped_type, Allocator, true >` Struct Template Reference

#### Public Types

- typedef mapped\_type\_allocator::const\_pointer **const\_mapped\_pointer**
- typedef mapped\_type\_allocator::const\_reference **const\_mapped\_reference**
- typedef value\_type\_allocator::const\_pointer **const\_pointer**
- typedef value\_type\_allocator::const\_reference **const\_reference**
- typedef mapped\_type\_allocator::pointer **mapped\_pointer**
- typedef mapped\_type\_allocator::reference **mapped\_reference**
- typedef mapped\_type\_allocator::value\_type **mapped\_type**
- typedef Allocator::template rebind< [null\\_mapped\\_type](#) >::other **mapped\_type\_allocator**
- typedef value\_type\_allocator::pointer **pointer**
- typedef value\_type\_allocator::reference **reference**
- typedef Key **value\_type**
- typedef Allocator::template rebind< value\_type >::other **value\_type\_allocator**

#### Static Public Attributes

- static [null\\_mapped\\_type](#) **s\_null\_mapped**

#### 5.192.1 Detailed Description

`template<typename Key, typename Allocator> struct __gnu_pbds::detail::value_type_base< Key, null_mapped_type, Allocator, true >`

Specialization of `value_type_base` for the case where the hash value is stored alongside each value.

Definition at line 164 of file `basic_types.hpp`.

The documentation for this struct was generated from the following file:

- [basic\\_types.hpp](#)

### 5.193 `__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, Allocator >` Class Template Reference

A concrete general-probing hash-based associative container.

**5.193 `__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, Allocator >` Class Template Reference** 1329

Inheritance diagram for `__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, Allocator >`:



**Public Types**

- typedef Allocator **allocator\_type**
- typedef Comb\_Probe\_Fn **comb\_probe\_fn**
- typedef base\_type::const\_iterator **const\_iterator**
- typedef key\_rebind::const\_pointer **const\_key\_pointer**
- typedef key\_rebind::const\_reference **const\_key\_reference**
- typedef mapped\_rebind::const\_pointer **const\_mapped\_pointer**
- typedef mapped\_rebind::const\_reference **const\_mapped\_reference**
- typedef base\_type::const\_point\_iterator **const\_point\_iterator**
- typedef value\_rebind::const\_pointer **const\_pointer**
- typedef value\_rebind::const\_reference **const\_reference**
- typedef [gp\\_hash\\_tag](#) **container\_category**
- typedef allocator\_type::difference\_type **difference\_type**
- typedef Eq\_Fn **eq\_fn**
- typedef Hash\_Fn **hash\_fn**
- typedef base\_type::iterator **iterator**
- typedef key\_rebind::pointer **key\_pointer**
- typedef allocator\_type::template rebind< key\_type >::other **key\_rebind**
- typedef key\_rebind::reference **key\_reference**
- typedef allocator\_type::template rebind< Key >::other::value\_type **key\_type**
- typedef mapped\_rebind::pointer **mapped\_pointer**
- typedef allocator\_type::template rebind< mapped\_type >::other **mapped\_rebind**
- typedef mapped\_rebind::reference **mapped\_reference**
- typedef Mapped **mapped\_type**
- typedef base\_type::point\_iterator **point\_iterator**
- typedef value\_rebind::pointer **pointer**
- typedef Probe\_Fn **probe\_fn**
- typedef value\_rebind::reference **reference**
- typedef Resize\_Policy **resize\_policy**
- typedef allocator\_type::size\_type **size\_type**
- typedef allocator\_type::template rebind< value\_type >::other **value\_rebind**
- typedef base\_type::value\_type **value\_type**

## Public Member Functions

- `gp_hash_table` (const hash\_fn &h)
- `gp_hash_table` (const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp)
- `template<typename It >`  
`gp_hash_table` (It first, It last, const hash\_fn &h)
- `gp_hash_table` (const [gp\\_hash\\_table](#) &other)
- `template<typename It >`  
`gp_hash_table` (It first, It last, const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp, const probe\_fn &p, const resize\_policy &rp)
- `template<typename It >`  
`gp_hash_table` (It first, It last, const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp, const probe\_fn &p)
- `template<typename It >`  
`gp_hash_table` (It first, It last, const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp)
- `template<typename It >`  
`gp_hash_table` (It first, It last, const hash\_fn &h, const eq\_fn &e)
- `gp_hash_table` (const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp, const probe\_fn &p)
- `template<typename It >`  
`gp_hash_table` (It first, It last)
- `gp_hash_table` (const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp, const probe\_fn &p, const resize\_policy &rp)
- `gp_hash_table` (const hash\_fn &h, const eq\_fn &e)
- `gp_hash_table & operator=` (const [gp\\_hash\\_table](#) &other)
- `void swap` ([gp\\_hash\\_table](#) &other)

### 5.193.1 Detailed Description

```
template<typename Key, typename Mapped, typename Hash_Fn = type-
name detail::default_hash_fn<Key>::type, typename Eq_Fn = type-
name detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename
detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy
= typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_
Hash = detail::default_store_hash, typename Allocator = std::allocator<char>>
class __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_
Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, Allocator >
```

A concrete general-probing hash-based associative container.

Definition at line 318 of file `assoc_container.hpp`.

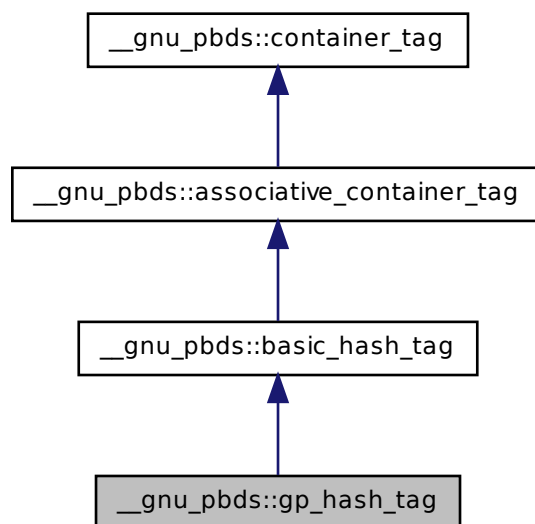
The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

## 5.194 `__gnu_pbds::gp_hash_tag` Struct Reference

General-probing hash.

Inheritance diagram for `__gnu_pbds::gp_hash_tag`:



### 5.194.1 Detailed Description

General-probing hash.

Definition at line 112 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.195 `__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, Allocator >` Class Template Reference

A list-update based associative container.

Inheritance diagram for `__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, Allocator >`:



### Public Types

- typedef Allocator **allocator**
- typedef Allocator **allocator\_type**
- typedef base\_type::const\_iterator **const\_iterator**
- typedef key\_rebind::const\_pointer **const\_key\_pointer**
- typedef key\_rebind::const\_reference **const\_key\_reference**
- typedef mapped\_rebind::const\_pointer **const\_mapped\_pointer**
- typedef mapped\_rebind::const\_reference **const\_mapped\_reference**
- typedef base\_type::const\_point\_iterator **const\_point\_iterator**
- typedef value\_rebind::const\_pointer **const\_pointer**
- typedef value\_rebind::const\_reference **const\_reference**
- typedef [list\\_update\\_tag](#) **container\_category**
- typedef allocator\_type::difference\_type **difference\_type**
- typedef Eq\_Fn **eq\_fn**
- typedef base\_type::iterator **iterator**
- typedef key\_rebind::pointer **key\_pointer**
- typedef allocator\_type::template rebind< key\_type >::other **key\_rebind**
- typedef key\_rebind::reference **key\_reference**
- typedef allocator\_type::template rebind< Key >::other::value\_type **key\_type**
- typedef mapped\_rebind::pointer **mapped\_pointer**
- typedef allocator\_type::template rebind< mapped\_type >::other **mapped\_rebind**
- typedef mapped\_rebind::reference **mapped\_reference**
- typedef Mapped **mapped\_type**
- typedef base\_type::point\_iterator **point\_iterator**
- typedef value\_rebind::pointer **pointer**
- typedef value\_rebind::reference **reference**
- typedef allocator\_type::size\_type **size\_type**
- typedef Update\_Policy **update\_policy**
- typedef allocator\_type::template rebind< value\_type >::other **value\_rebind**
- typedef base\_type::value\_type **value\_type**

### Public Member Functions

- `template<typename It >`  
`list_update` (It first, It last)
- `list_update` (const [list\\_update](#) &other)
- `list_update` & `operator=` (const [list\\_update](#) &other)
- `void swap` ([list\\_update](#) &other)

#### 5.195.1 Detailed Description

```
template<typename Key, typename Mapped, class Eq_Fn = typename
detail::default_eq_fn<Key>::type, class Update_Policy = detail::default_
update_policy::type, class Allocator = std::allocator<char>> class __gnu_
pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, Allocator >
```

A list-update based associative container.

Definition at line 654 of file `assoc_container.hpp`.

The documentation for this class was generated from the following file:

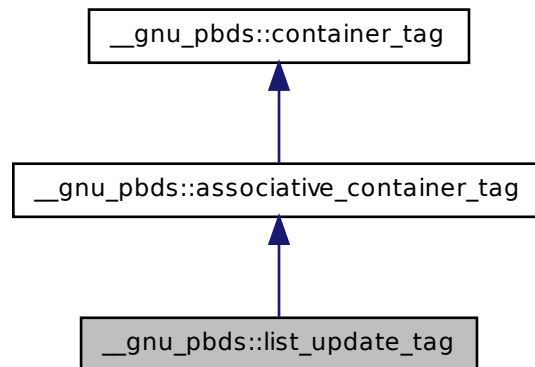
- [assoc\\_container.hpp](#)

### 5.196 `__gnu_pbds::list_update_tag` Struct Reference

List-update.



Inheritance diagram for `__gnu_pbds::list_update_tag`:



#### 5.196.1 Detailed Description

List-update.

Definition at line 136 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 5.197 `__gnu_pbds::null_mapped_type` Struct Reference

A mapped-policy indicating that an associative container is a set.

#### 5.197.1 Detailed Description

A mapped-policy indicating that an associative container is a set.

Definition at line 89 of file `tag_and_trait.hpp`.

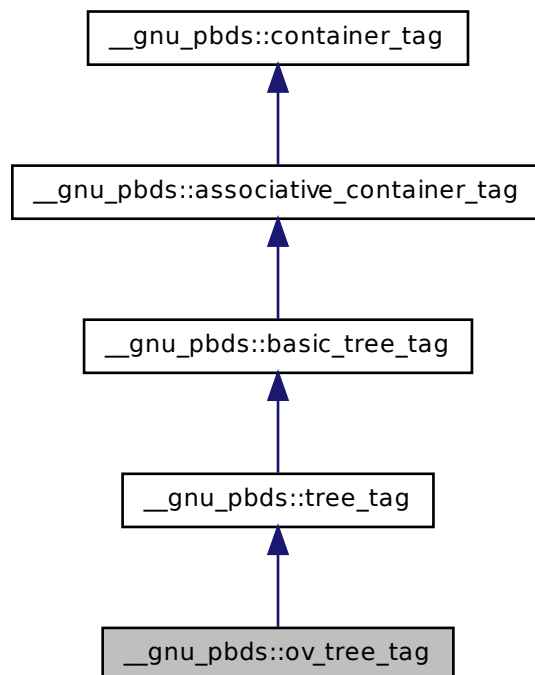
The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.198 \_\_gnu\_pbds::ov\_tree\_tag Struct Reference

Ordered-vector tree.

Inheritance diagram for \_\_gnu\_pbds::ov\_tree\_tag:



### 5.198.1 Detailed Description

Ordered-vector tree.

Definition at line 127 of file `tag_and_trait.hpp`.

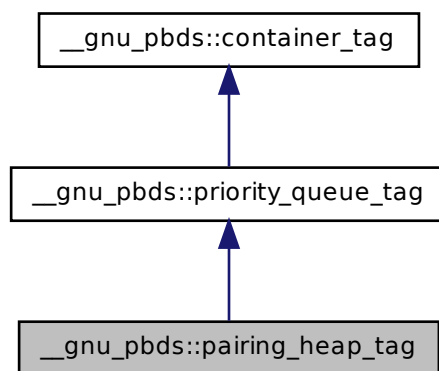
The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.199 \_\_gnu\_pbds::pairing\_heap\_tag Struct Reference

Pairing-heap.

Inheritance diagram for \_\_gnu\_pbds::pairing\_heap\_tag:



### 5.199.1 Detailed Description

Pairing-heap.

Definition at line 142 of file `tag_and_trait.hpp`.

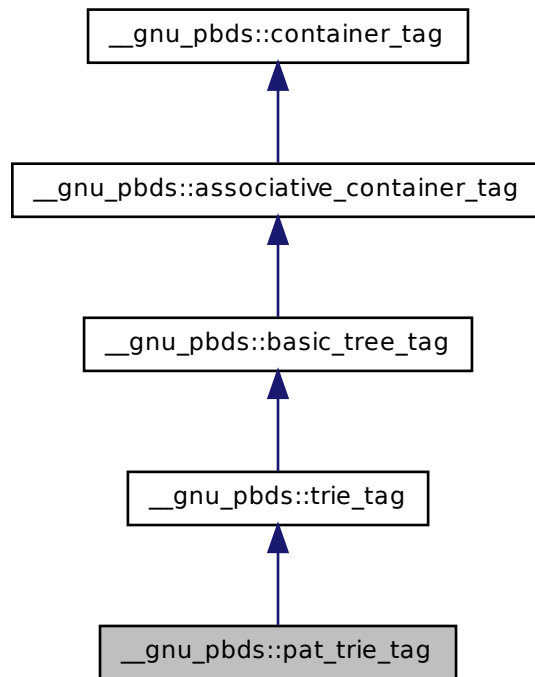
The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.200 \_\_gnu\_pbds::pat\_trie\_tag Struct Reference

PATRICIA trie.

Inheritance diagram for \_\_gnu\_pbds::pat\_trie\_tag:



#### 5.200.1 Detailed Description

PATRICIA trie.

Definition at line 133 of file `tag_and_trait.hpp`.

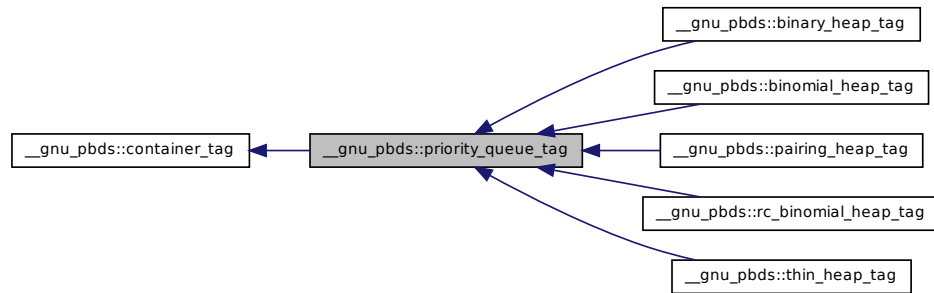
The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 5.201 \_\_gnu\_pbds::priority\_queue\_tag Struct Reference

Basic priority-queue.

Inheritance diagram for \_\_gnu\_pbds::priority\_queue\_tag:



### 5.201.1 Detailed Description

Basic priority-queue.

Definition at line 139 of file `tag_and_trait.hpp`.

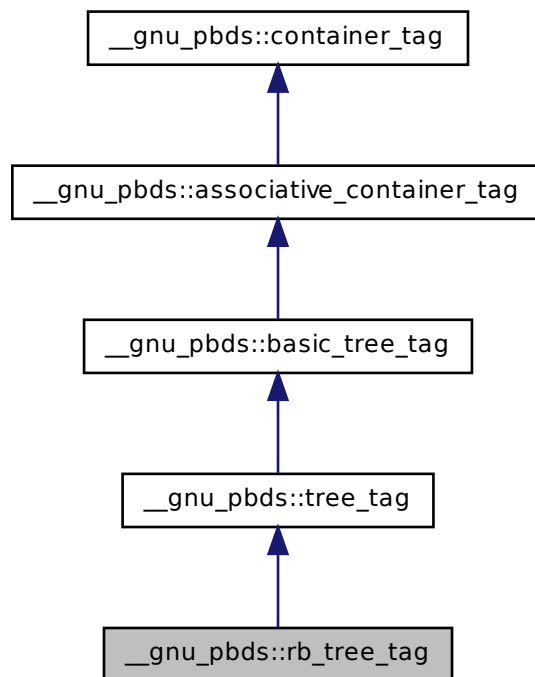
The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 5.202 \_\_gnu\_pbds::rb\_tree\_tag Struct Reference

Red-black tree.

Inheritance diagram for `__gnu_pbds::rb_tree_tag`:



#### 5.202.1 Detailed Description

Red-black tree.

Definition at line 121 of file `tag_and_trait.hpp`.

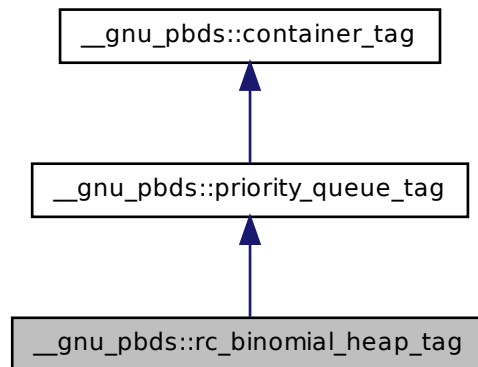
The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 5.203 `__gnu_pbds::rc_binomial_heap_tag` Struct Reference

Redundant-counter binomial-heap.

Inheritance diagram for \_\_gnu\_pbds::rc\_binomial\_heap\_tag:



#### 5.203.1 Detailed Description

Redundant-counter binomial-heap.

Definition at line 148 of file `tag_and_trait.hpp`.

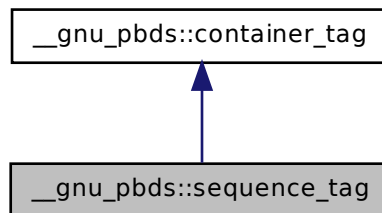
The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 5.204 \_\_gnu\_pbds::sequence\_tag Struct Reference

Basic sequence.

Inheritance diagram for \_\_gnu\_pbds::sequence\_tag:



#### 5.204.1 Detailed Description

Basic sequence.

Definition at line 100 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

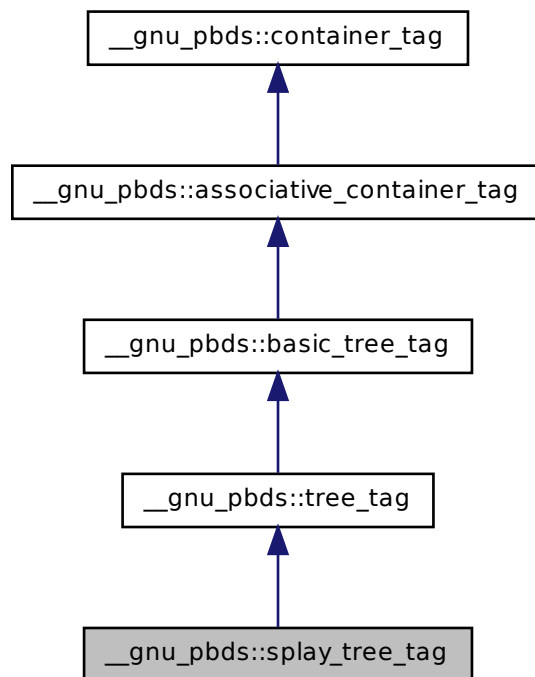
- [tag\\_and\\_trait.hpp](#)

#### 5.205 \_\_gnu\_pbds::splay\_tree\_tag Struct Reference

Splay tree.



Inheritance diagram for \_\_gnu\_pbds::splay\_tree\_tag:



#### 5.205.1 Detailed Description

Splay tree.

Definition at line 124 of file `tag_and_trait.hpp`.

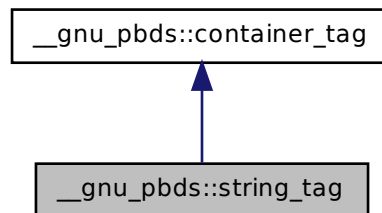
The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 5.206 \_\_gnu\_pbds::string\_tag Struct Reference

Basic string container, inclusive of strings, ropes, etc.

Inheritance diagram for `__gnu_pbds::string_tag`:



#### 5.206.1 Detailed Description

Basic string container, inclusive of strings, ropes, etc.

Definition at line 97 of file `tag_and_trait.hpp`.

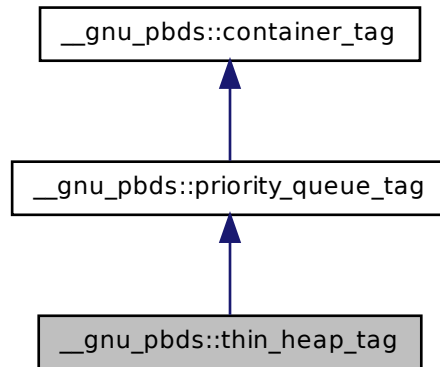
The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 5.207 `__gnu_pbds::thin_heap_tag` Struct Reference

Thin heap.

Inheritance diagram for `__gnu_pbds::thin_heap_tag`:



#### 5.207.1 Detailed Description

Thin heap.

Definition at line 154 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 5.208 `__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, Allocator >` Class Template Reference

A concrete basic tree-based associative container.

Inheritance diagram for `__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, Allocator >`:



## Public Types

- typedef Allocator **allocator\_type**
- typedef Cmp\_Fn **cmp\_fn**
- typedef base\_type::const\_iterator **const\_iterator**
- typedef key\_rebind::const\_pointer **const\_key\_pointer**
- typedef key\_rebind::const\_reference **const\_key\_reference**
- typedef mapped\_rebind::const\_pointer **const\_mapped\_pointer**
- typedef mapped\_rebind::const\_reference **const\_mapped\_reference**
- typedef base\_type::const\_point\_iterator **const\_point\_iterator**
- typedef value\_rebind::const\_pointer **const\_pointer**
- typedef value\_rebind::const\_reference **const\_reference**
- typedef Tag **container\_category**
- typedef allocator\_type::difference\_type **difference\_type**
- typedef base\_type::iterator **iterator**
- typedef key\_rebind::pointer **key\_pointer**
- typedef allocator\_type::template rebind< key\_type >::other **key\_rebind**
- typedef key\_rebind::reference **key\_reference**
- typedef allocator\_type::template rebind< Key >::other::value\_type **key\_type**
- typedef mapped\_rebind::pointer **mapped\_pointer**
- typedef allocator\_type::template rebind< mapped\_type >::other **mapped\_rebind**
- typedef mapped\_rebind::reference **mapped\_reference**
- typedef Mapped **mapped\_type**
- typedef detail::tree\_traits< Key, Mapped, Cmp\_Fn, Node\_Update, Tag, Allocator >::node\_update **node\_update**
- typedef base\_type::point\_iterator **point\_iterator**
- typedef value\_rebind::pointer **pointer**
- typedef value\_rebind::reference **reference**
- typedef allocator\_type::size\_type **size\_type**
- typedef allocator\_type::template rebind< value\_type >::other **value\_rebind**
- typedef base\_type::value\_type **value\_type**

## Public Member Functions

- **tree** (const cmp\_fn &c)
- template<typename It >  
  **tree** (It first, It last, const cmp\_fn &c)
- **tree** (const [tree](#) &other)
- template<typename It >  
  **tree** (It first, It last)
- [tree](#) & **operator=** (const [tree](#) &other)
- void **swap** ([tree](#) &other)

### 5.208.1 Detailed Description

```
template<typename Key, typename Mapped, typename Cmp_Fn =
std::less<Key>, typename Tag = rb_tree_tag, template< typename Const_
Node_Iterator, typename Node_Iterator, typename Cmp_Fn_, typename
Allocator_ > class Node_Update = __gnu_pbds::null_tree_node_update, type-
name Allocator = std::allocator<char>> class __gnu_pbds::tree< Key, Mapped,
Cmp_Fn, Tag, Node_Update, Allocator >
```

A concrete basic tree-based associative container.

Definition at line 510 of file `assoc_container.hpp`.

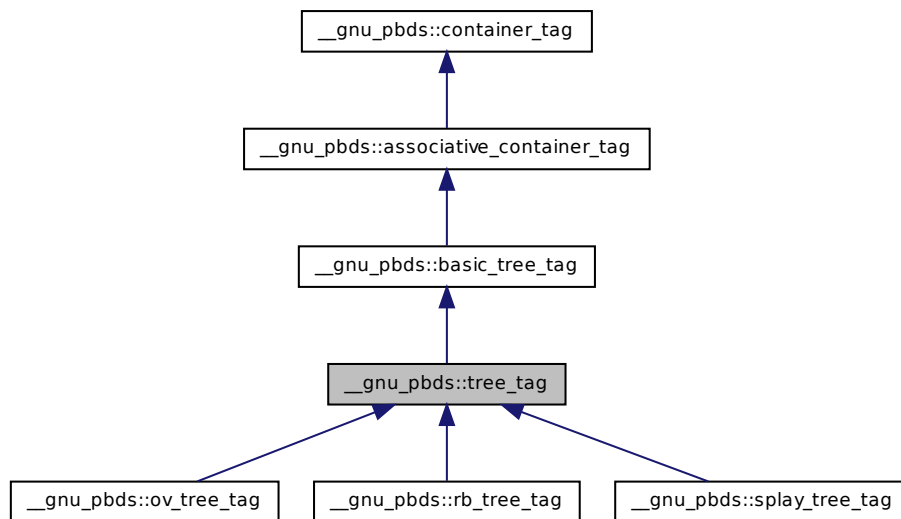
The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

## 5.209 \_\_gnu\_pbds::tree\_tag Struct Reference

tree.

Inheritance diagram for `__gnu_pbds::tree_tag`:



### 5.209.1 Detailed Description

tree.

Definition at line 118 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 5.210 `__gnu_pbds::trie< Key, Mapped, E_Access_Traits, Tag, Node_Update, Allocator >` Class Template Reference

A concrete basic trie-based associative container.

Inheritance diagram for `__gnu_pbds::trie< Key, Mapped, E_Access_Traits, Tag, Node_Update, Allocator >`:



#### Public Types

- typedef Allocator **allocator\_type**
- typedef base\_type::const\_iterator **const\_iterator**
- typedef key\_rebind::const\_pointer **const\_key\_pointer**
- typedef key\_rebind::const\_reference **const\_key\_reference**
- typedef mapped\_rebind::const\_pointer **const\_mapped\_pointer**
- typedef mapped\_rebind::const\_reference **const\_mapped\_reference**
- typedef base\_type::const\_point\_iterator **const\_point\_iterator**
- typedef value\_rebind::const\_pointer **const\_pointer**
- typedef value\_rebind::const\_reference **const\_reference**
- typedef Tag **container\_category**
- typedef allocator\_type::difference\_type **difference\_type**
- typedef E\_Access\_Traits **e\_access\_traits**
- typedef base\_type::iterator **iterator**
- typedef key\_rebind::pointer **key\_pointer**
- typedef allocator\_type::template rebind< key\_type >::other **key\_rebind**
- typedef key\_rebind::reference **key\_reference**
- typedef allocator\_type::template rebind< Key >::other::value\_type **key\_type**
- typedef mapped\_rebind::pointer **mapped\_pointer**
- typedef allocator\_type::template rebind< mapped\_type >::other **mapped\_rebind**
- typedef mapped\_rebind::reference **mapped\_reference**

- typedef Mapped **mapped\_type**
- typedef detail::trie\_traits< Key, Mapped, E\_Access\_Traits, Node\_Update, Tag, Allocator >::node\_update **node\_update**
- typedef base\_type::point\_iterator **point\_iterator**
- typedef value\_rebind::pointer **pointer**
- typedef value\_rebind::reference **reference**
- typedef allocator\_type::size\_type **size\_type**
- typedef allocator\_type::template rebind< value\_type >::other **value\_rebind**
- typedef base\_type::value\_type **value\_type**

### Public Member Functions

- **trie** (const e\_access\_traits &t)
- template<typename It >  
  **trie** (It first, It last, const e\_access\_traits &t)
- **trie** (const [trie](#) &other)
- template<typename It >  
  **trie** (It first, It last)
- [trie](#) & **operator=** (const [trie](#) &other)
- void **swap** ([trie](#) &other)

#### 5.210.1 Detailed Description

template<typename Key, typename Mapped, typename E\_Access\_Traits = typename detail::default\_trie\_e\_access\_traits<Key>::type, typename Tag = pat\_trie\_tag, template< typename Const\_Node\_Iterator, typename Node\_Iterator, typename E\_Access\_Traits\_, typename Allocator\_ > class Node\_Update = null\_trie\_node\_update, typename Allocator = std::allocator<char>> class \_\_gnu\_pbds::trie< Key, Mapped, E\_Access\_Traits, Tag, Node\_Update, Allocator >

A concrete basic trie-based associative container.

Definition at line 586 of file assoc\_container.hpp.

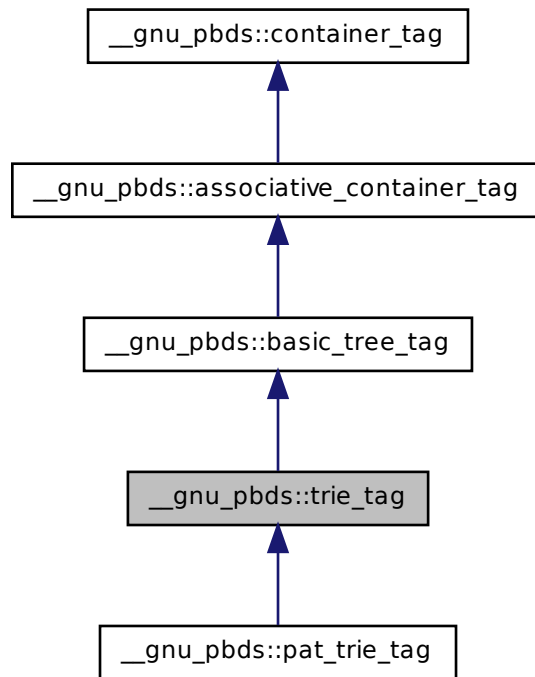
The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

### 5.211 \_\_gnu\_pbds::trie\_tag Struct Reference

trie.

Inheritance diagram for \_\_gnu\_pbds::trie\_tag:



#### 5.211.1 Detailed Description

trie.

Definition at line 130 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

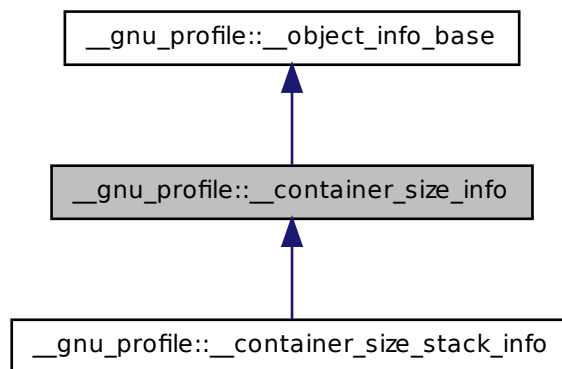
- [tag\\_and\\_trait.hpp](#)

#### 5.212 \_\_gnu\_profile::\_\_container\_size\_info Class Reference

A container size instrumentation line in the object table.



Inheritance diagram for `__gnu_profile::__container_size_info`:



### Public Member Functions

- `__container_size_info` (const `__container_size_info` &\_\_o)
- `__container_size_info` (`__stack_t` \_\_stack, std::size\_t \_\_num)
- `std::string` `__advice` () const
- void `__destruct` (std::size\_t \_\_num, std::size\_t \_\_inum)
- bool `__is_valid` () const
- float `__magnitude` () const
- void `__merge` (const `__container_size_info` &\_\_o)
- void `__resize` (std::size\_t \_\_from, std::size\_t \_\_to)
- float `__resize_cost` (std::size\_t \_\_from, std::size\_t)
- `__stack_t` `__stack` () const
- void `__write` (FILE \*\_\_f) const

### Protected Attributes

- `__stack_t` `_M_stack`
- bool `_M_valid`

### 5.212.1 Detailed Description

A container size instrumentation line in the object table.

Definition at line 49 of file `profiler_container_size.h`.

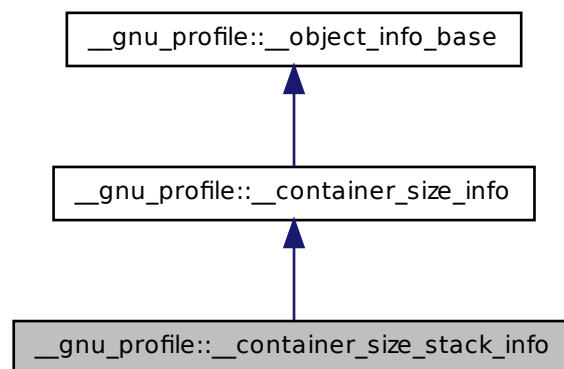
The documentation for this class was generated from the following file:

- [profiler\\_container\\_size.h](#)

## 5.213 `__gnu_profile::__container_size_stack_info` Class Reference

A container size instrumentation line in the stack table.

Inheritance diagram for `__gnu_profile::__container_size_stack_info`:



### Public Member Functions

- `__container_size_stack_info` (const [\\_\\_container\\_size\\_info](#) &\_\_o)
- [std::string](#) `__advice` () const
- void `__destruct` (std::size\_t \_\_num, std::size\_t \_\_inum)
- bool `__is_valid` () const
- float `__magnitude` () const
- void `__merge` (const [\\_\\_container\\_size\\_info](#) &\_\_o)
- void `__resize` (std::size\_t \_\_from, std::size\_t \_\_to)

- float **\_\_resize\_cost** (std::size\_t \_\_from, std::size\_t)
- \_\_stack\_t **\_\_stack** () const
- void **\_\_write** (FILE \*\_\_f) const

#### Protected Attributes

- \_\_stack\_t **\_M\_stack**
- bool **\_M\_valid**

##### 5.213.1 Detailed Description

A container size instrumentation line in the stack table.

Definition at line 161 of file profiler\_container\_size.h.

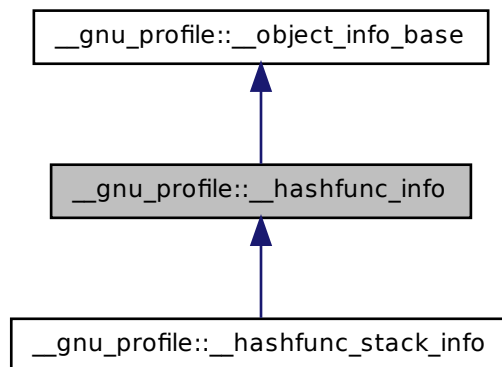
The documentation for this class was generated from the following file:

- [profiler\\_container\\_size.h](#)

#### 5.214 \_\_gnu\_profile::\_\_hashfunc\_info Class Reference

A hash performance instrumentation line in the object table.

Inheritance diagram for \_\_gnu\_profile::\_\_hashfunc\_info:



### Public Member Functions

- `__hashfunc_info` (const `__hashfunc_info` &\_\_o)
- `__hashfunc_info` (`__stack_t` \_\_stack)
- `std::string __advice` () const
- void `__destruct` (std::size\_t \_\_chain, std::size\_t \_\_accesses, std::size\_t \_\_hops)
- bool `__is_valid` () const
- float `__magnitude` () const
- void `__merge` (const `__hashfunc_info` &\_\_o)
- `__stack_t __stack` () const
- void `__write` (FILE \*\_\_f) const

### Protected Attributes

- `__stack_t _M_stack`
- bool `_M_valid`

#### 5.214.1 Detailed Description

A hash performance instrumentation line in the object table.

Definition at line 47 of file `profiler_hash_func.h`.

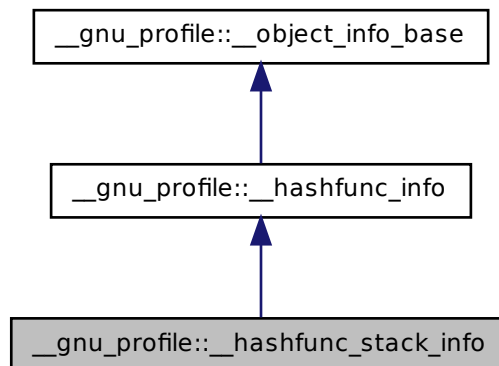
The documentation for this class was generated from the following file:

- `profiler_hash_func.h`

### 5.215 `__gnu_profile::__hashfunc_stack_info` Class Reference

A hash performance instrumentation line in the stack table.

Inheritance diagram for `__gnu_profile::__hashfunc_stack_info`:



### Public Member Functions

- `__hashfunc_stack_info` (const `__hashfunc_info` &\_\_o)
- `std::string __advice` () const
- void `__destruct` (std::size\_t \_\_chain, std::size\_t \_\_accesses, std::size\_t \_\_hops)
- bool `__is_valid` () const
- float `__magnitude` () const
- void `__merge` (const `__hashfunc_info` &\_\_o)
- `__stack_t __stack` () const
- void `__write` (FILE \*\_\_f) const

### Protected Attributes

- `__stack_t _M_stack`
- bool `_M_valid`

#### 5.215.1 Detailed Description

A hash performance instrumentation line in the stack table.

Definition at line 102 of file `profiler_hash_func.h`.

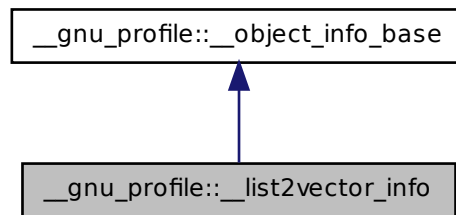
The documentation for this class was generated from the following file:

- [profiler\\_hash\\_func.h](#)

## 5.216 \_\_gnu\_profile::\_\_list2vector\_info Class Reference

A list-to-vector instrumentation line in the object table.

Inheritance diagram for \_\_gnu\_profile::\_\_list2vector\_info:



### Public Member Functions

- `__list2vector_info` (`__stack_t __stack`)
- `__list2vector_info` (`const __list2vector_info &__o`)
- `std::string __advice` () const
- `bool __is_valid` () const
- `bool __is_valid` ()
- `std::size_t __iterate` ()
- `float __list_cost` ()
- `float __magnitude` () const
- `void __merge` (`const __list2vector_info &__o`)
- `void __opr_insert` (`std::size_t __shift`, `std::size_t __size`)
- `void __opr_iterate` (`std::size_t __num`)
- `std::size_t __resize` ()
- `void __resize` (`std::size_t __from`, `std::size_t`)
- `void __set_invalid` ()
- `void __set_list_cost` (`float __lc`)
- `void __set_vector_cost` (`float __vc`)
- `std::size_t __shift_count` ()
- `__stack_t __stack` () const
- `void __write` (`FILE *__f`) const

**Protected Attributes**

- `__stack_t _M_stack`

**5.216.1 Detailed Description**

A list-to-vector instrumentation line in the object table.

Definition at line 49 of file `profiler_list_to_vector.h`.

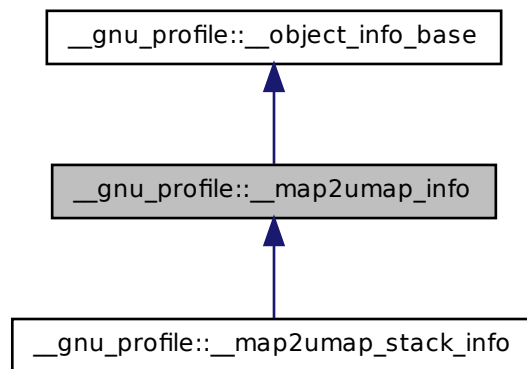
The documentation for this class was generated from the following file:

- [profiler\\_list\\_to\\_vector.h](#)

**5.217 \_\_gnu\_profile::\_\_map2umap\_info Class Reference**

A map-to-unordered\_map instrumentation line in the object table.

Inheritance diagram for `__gnu_profile::__map2umap_info`:

**Public Member Functions**

- `__map2umap_info (__stack_t __stack)`
- `__map2umap_info (const __map2umap_info &__o)`
- `std::string __advice () const`

- `bool __is_valid () const`
- `float __magnitude () const`
- `void __merge (const \_\_map2umap\_info &__o)`
- `void __record_erase (std::size_t __size, std::size_t __count)`
- `void __record_find (std::size_t __size)`
- `void __record_insert (std::size_t __size, std::size_t __count)`
- `void __record_invalidate ()`
- `void __record_iterate (std::size_t __count)`
- `__stack_t __stack () const`
- `void __write (FILE *__f) const`

#### Protected Attributes

- `__stack_t _M_stack`

#### 5.217.1 Detailed Description

A map-to-unordered\_map instrumentation line in the object table.

Definition at line 73 of file `profiler_map_to_unordered_map.h`.

The documentation for this class was generated from the following file:

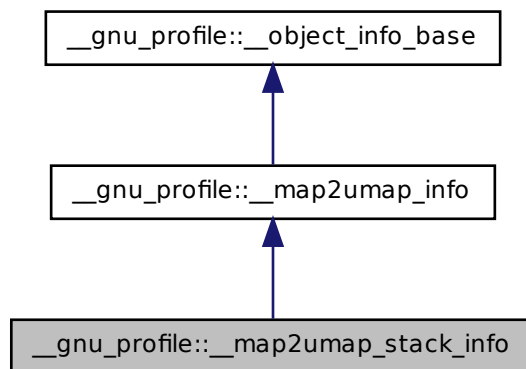
- [profiler\\_map\\_to\\_unordered\\_map.h](#)

#### 5.218 `__gnu_profile::__map2umap_stack_info` Class Reference

A map-to-unordered\_map instrumentation line in the stack table.



Inheritance diagram for \_\_gnu\_profile::\_\_map2umap\_stack\_info:



### Public Member Functions

- [\\_\\_map2umap\\_stack\\_info](#) (const [\\_\\_map2umap\\_info](#) &\_\_o)
- [std::string \\_\\_advice](#) () const
- bool [\\_\\_is\\_valid](#) () const
- float [\\_\\_magnitude](#) () const
- void [\\_\\_merge](#) (const [\\_\\_map2umap\\_info](#) &\_\_o)
- void [\\_\\_record\\_erase](#) (std::size\_t \_\_size, std::size\_t \_\_count)
- void [\\_\\_record\\_find](#) (std::size\_t \_\_size)
- void [\\_\\_record\\_insert](#) (std::size\_t \_\_size, std::size\_t \_\_count)
- void [\\_\\_record\\_invalidate](#) ()
- void [\\_\\_record\\_iterate](#) (std::size\_t \_\_count)
- [\\_\\_stack\\_t \\_\\_stack](#) () const
- void [\\_\\_write](#) (FILE \*\_\_f) const

### Protected Attributes

- [\\_\\_stack\\_t \\_M\\_stack](#)

### 5.218.1 Detailed Description

A map-to-unordered\_map instrumentation line in the stack table.

Definition at line 177 of file `profiler_map_to_unordered_map.h`.

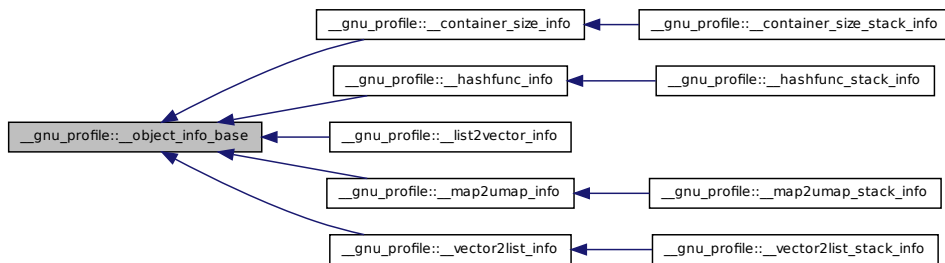
The documentation for this class was generated from the following file:

- [profiler\\_map\\_to\\_unordered\\_map.h](#)

## 5.219 `__gnu_profile::__object_info_base` Class Reference

Base class for a line in the object table.

Inheritance diagram for `__gnu_profile::__object_info_base`:



### Public Member Functions

- `__object_info_base` (`__stack_t` `__stack`)
- `__object_info_base` (`const` `__object_info_base` &`__o`)
- `bool` `__is_valid` () `const`
- `__stack_t` `__stack` () `const`
- `virtual void` `__write` (`FILE *` `__f`) `const` `=0`

### Protected Attributes

- `__stack_t` `_M_stack`
- `bool` `_M_valid`

### 5.219.1 Detailed Description

Base class for a line in the object table.

Definition at line 130 of file `profiler_node.h`.

The documentation for this class was generated from the following file:

- [profiler\\_node.h](#)

## 5.220 `__gnu_profile::__reentrance_guard` Struct Reference

Reentrance guard.

### Static Public Member Functions

- static bool `__get_in()`
- static bool & `__inside()`

### 5.220.1 Detailed Description

Reentrance guard. Mechanism to protect all [\\_\\_gnu\\_profile](#) operations against recursion, multithreaded and exception reentrance.

Definition at line 65 of file `profiler.h`.

The documentation for this struct was generated from the following file:

- [profiler.h](#)

## 5.221 `__gnu_profile::__stack_hash` Class Reference

Hash function for summary trace using call stack as index.

### Public Member Functions

- `std::size_t operator() (__stack_t __s) const`
- `bool operator() (__stack_t __stack1, __stack_t __stack2) const`

### 5.221.1 Detailed Description

Hash function for summary trace using call stack as index.

Definition at line 96 of file `profiler_node.h`.

The documentation for this class was generated from the following file:

- [profiler\\_node.h](#)

## 5.222 `__gnu_profile::__stack_info_base< __object_info >` Class Template Reference

Base class for a line in the stack table.

### Public Member Functions

- `__stack_info_base` (const `__object_info` &`__info`)=0
- virtual const char \* `__get_id` () const =0
- virtual float `__magnitude` () const =0
- void `__merge` (const `__object_info` &`__info`)=0

#### 5.222.1 Detailed Description

template<typename `__object_info`> class `__gnu_profile::__stack_info_base< __object_info >`

Base class for a line in the stack table.

Definition at line 161 of file `profiler_node.h`.

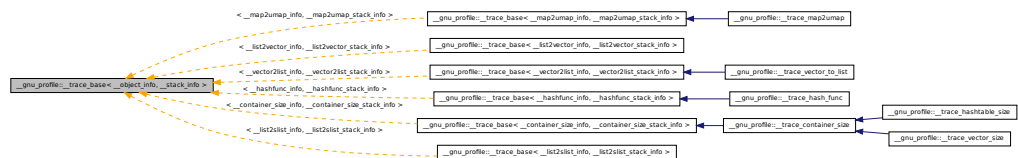
The documentation for this class was generated from the following file:

- [profiler\\_node.h](#)

## 5.223 `__gnu_profile::__trace_base< __object_info, __stack_info >` Class Template Reference

Base class for all trace producers.

Inheritance diagram for `__gnu_profile::__trace_base< __object_info, __stack_info >`:



### Public Member Functions

- void **\_\_add\_object** (`__object_t` object, `__object_info` \_\_info)
- void **\_\_collect\_warnings** (`__warning_vector_t` &\_\_warnings)
- `__object_info` \* **\_\_get\_object\_info** (`__object_t` \_\_object)
- void **\_\_retire\_object** (`__object_t` \_\_object)
- void **\_\_write** (`FILE` \*\_\_f)

### Protected Attributes

- const char \* **\_\_id**

#### 5.223.1 Detailed Description

template<typename `__object_info`, typename `__stack_info`> class `__gnu_profile::__trace_base`< `__object_info`, `__stack_info` >

Base class for all trace producers.

Definition at line 190 of file `profiler_trace.h`.

The documentation for this class was generated from the following file:

- [profiler\\_trace.h](#)

## 5.224 `__gnu_profile::__trace_container_size` Class Reference

Container size instrumentation trace producer.

Inheritance diagram for `__gnu_profile::__trace_container_size`:



### Public Member Functions

- void **\_\_add\_object** (`__object_t` object, `__container_size_info` \_\_info)
- void **\_\_collect\_warnings** (`__warning_vector_t` &\_\_warnings)
- void **\_\_construct** (const void \* \_\_obj, std::size\_t \_\_inum)
- void **\_\_destruct** (const void \* \_\_obj, std::size\_t \_\_num, std::size\_t \_\_inum)
- `__container_size_info` \* **\_\_get\_object\_info** (`__object_t` \_\_object)
- void **\_\_insert** (const `__object_t` \_\_obj, `__stack_t` \_\_stack, std::size\_t \_\_num)

- void **\_\_resize** (const void \*\_\_obj, int \_\_from, int \_\_to)
- void **\_\_retire\_object** (\_\_object\_t \_\_object)
- void **\_\_write** (FILE \*\_\_f)

### Protected Attributes

- const char \* **\_\_id**

#### 5.224.1 Detailed Description

Container size instrumentation trace producer.

Definition at line 171 of file profiler\_container\_size.h.

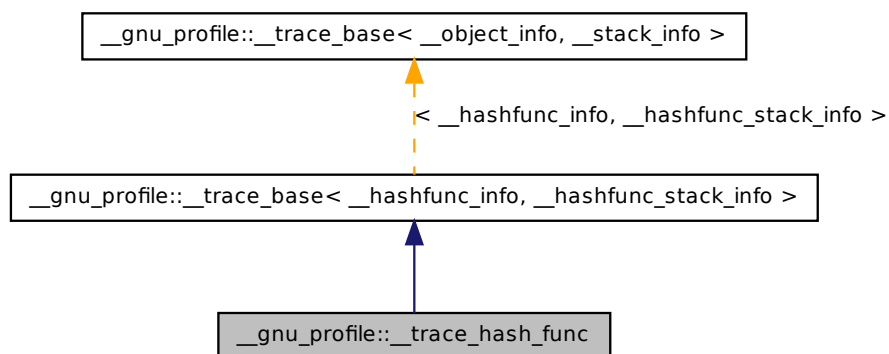
The documentation for this class was generated from the following file:

- [profiler\\_container\\_size.h](#)

#### 5.225 \_\_gnu\_profile::\_\_trace\_hash\_func Class Reference

Hash performance instrumentation producer.

Inheritance diagram for \_\_gnu\_profile::\_\_trace\_hash\_func:



**Public Member Functions**

- void **\_\_add\_object** (`__object_t` object, `__hashfunc_info__info`)
- void **\_\_collect\_warnings** (`__warning_vector_t` &`__warnings`)
- void **\_\_destruct** (const void \*`__obj`, `std::size_t` `__chain`, `std::size_t` `__accesses`, `std::size_t` `__hops`)
- `__hashfunc_info` \* **\_\_get\_object\_info** (`__object_t` `__object`)
- void **\_\_insert** (`__object_t` `__obj`, `__stack_t` `__stack`)
- void **\_\_retire\_object** (`__object_t` `__object`)
- void **\_\_write** (FILE \*`__f`)

**Protected Attributes**

- const char \* **\_\_id**

**5.225.1 Detailed Description**

Hash performance instrumentation producer.

Definition at line 112 of file `profiler_hash_func.h`.

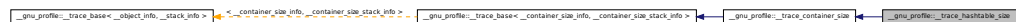
The documentation for this class was generated from the following file:

- [profiler\\_hash\\_func.h](#)

**5.226 `__gnu_profile::__trace_hashtable_size` Class Reference**

Hashtable size instrumentation trace producer.

Inheritance diagram for `__gnu_profile::__trace_hashtable_size`:

**Public Member Functions**

- void **\_\_add\_object** (`__object_t` object, `__container_size_info__info`)
- void **\_\_collect\_warnings** (`__warning_vector_t` &`__warnings`)
- void **\_\_construct** (const void \*`__obj`, `std::size_t` `__inum`)
- void **\_\_destruct** (const void \*`__obj`, `std::size_t` `__num`, `std::size_t` `__inum`)
- `__container_size_info` \* **\_\_get\_object\_info** (`__object_t` `__object`)
- void **\_\_insert** (const `__object_t` `__obj`, `__stack_t` `__stack`, `std::size_t` `__num`)

- void **\_\_resize** (const void \*\_\_obj, int \_\_from, int \_\_to)
- void **\_\_retire\_object** (\_\_object\_t \_\_object)
- void **\_\_write** (FILE \*\_\_f)

### Protected Attributes

- const char \* **\_\_id**

#### 5.226.1 Detailed Description

Hashtable size instrumentation trace producer.

Definition at line 49 of file profiler\_hashtable\_size.h.

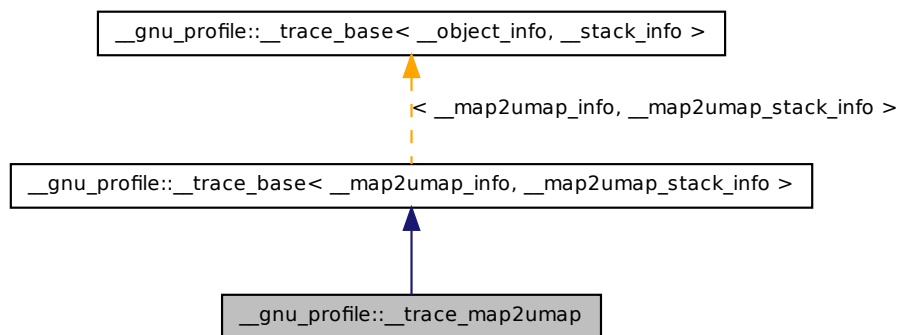
The documentation for this class was generated from the following file:

- [profiler\\_hashtable\\_size.h](#)

#### 5.227 \_\_gnu\_profile::\_\_trace\_map2umap Class Reference

Map-to-unordered\_map instrumentation producer.

Inheritance diagram for \_\_gnu\_profile::\_\_trace\_map2umap:



### Public Member Functions

- void **\_\_add\_object** (\_\_object\_t object, \_\_map2umap\_info\_\_info)



- void `__collect_warnings` (`__warning_vector_t` &`__warnings`)
- `__map2umap_info` \* `__get_object_info` (`__object_t` `__object`)
- void `__retire_object` (`__object_t` `__object`)
- void `__write` (`FILE` \*`__f`)

### Protected Attributes

- const char \* `__id`

#### 5.227.1 Detailed Description

Map-to-unordered\_map instrumentation producer.

Definition at line 186 of file `profiler_map_to_unordered_map.h`.

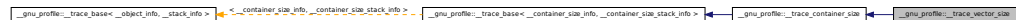
The documentation for this class was generated from the following file:

- [profiler\\_map\\_to\\_unordered\\_map.h](#)

## 5.228 `__gnu_profile::__trace_vector_size` Class Reference

Hashtable size instrumentation trace producer.

Inheritance diagram for `__gnu_profile::__trace_vector_size`:



### Public Member Functions

- void `__add_object` (`__object_t` `object`, `__container_size_info` `__info`)
- void `__collect_warnings` (`__warning_vector_t` &`__warnings`)
- void `__construct` (const void \*`__obj`, `std::size_t` `__inum`)
- void `__destruct` (const void \*`__obj`, `std::size_t` `__num`, `std::size_t` `__inum`)
- `__container_size_info` \* `__get_object_info` (`__object_t` `__object`)
- void `__insert` (const `__object_t` `__obj`, `__stack_t` `__stack`, `std::size_t` `__num`)
- void `__resize` (const void \*`__obj`, int `__from`, int `__to`)
- void `__retire_object` (`__object_t` `__object`)
- void `__write` (`FILE` \*`__f`)

### Protected Attributes

- const char \* `__id`

### 5.228.1 Detailed Description

Hashtable size instrumentation trace producer.

Definition at line 49 of file `profiler_vector_size.h`.

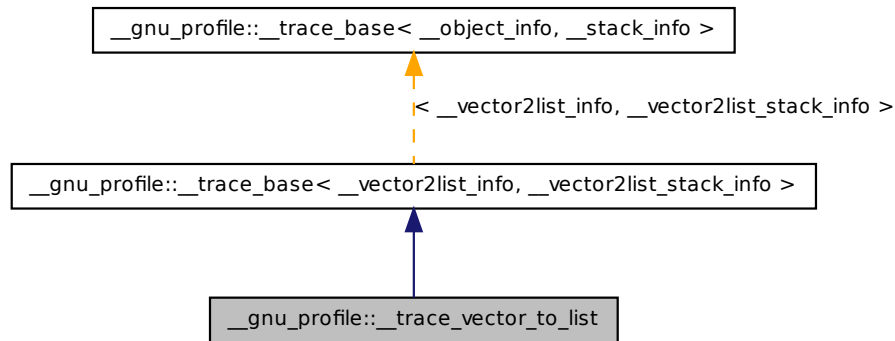
The documentation for this class was generated from the following file:

- [profiler\\_vector\\_size.h](#)

## 5.229 `__gnu_profile::__trace_vector_to_list` Class Reference

Vector-to-list instrumentation producer.

Inheritance diagram for `__gnu_profile::__trace_vector_to_list`:



### Public Member Functions

- `void __add_object (__object_t object, __vector2list_info __info)`
- `void __collect_warnings (__warning_vector_t &__warnings)`
- `void __destruct (const void *__obj)`
- `__vector2list_info * __find (const void *__obj)`
- `__vector2list_info * __get_object_info (__object_t __object)`
- `void __insert (__object_t __obj, __stack_t __stack)`
- `void __invalid_operator (const void *__obj)`
- `float __list_cost (std::size_t __shift, std::size_t __iterate, std::size_t __resize)`
- `void __opr_find (const void *__obj, std::size_t __size)`

- void `__opr_insert` (const void \*`__obj`, std::size\_t `__pos`, std::size\_t `__num`)
- void `__opr_iterate` (const void \*`__obj`, std::size\_t `__num`)
- void `__resize` (const void \*`__obj`, std::size\_t `__from`, std::size\_t `__to`)
- void `__retire_object` (`__object_t` `__object`)
- float `__vector_cost` (std::size\_t `__shift`, std::size\_t `__iterate`, std::size\_t `__resize`)
- void `__write` (FILE \*`__f`)

#### Protected Attributes

- const char \* `__id`

##### 5.229.1 Detailed Description

Vector-to-list instrumentation producer.

Definition at line 165 of file `profiler_vector_to_list.h`.

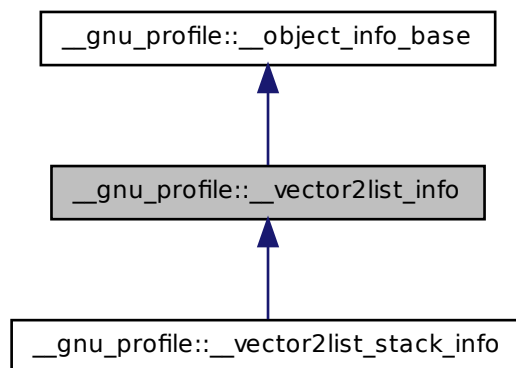
The documentation for this class was generated from the following file:

- [profiler\\_vector\\_to\\_list.h](#)

##### 5.230 `__gnu_profile::__vector2list_info` Class Reference

A vector-to-list instrumentation line in the object table.

Inheritance diagram for `__gnu_profile::__vector2list_info`:



### Public Member Functions

- `__vector2list_info` (`__stack_t __stack`)
- `__vector2list_info` (`const __vector2list_info &__o`)
- `std::string __advice` () const
- `bool __is_valid` () const
- `bool __is_valid` ()
- `std::size_t __iterate` ()
- `float __list_cost` ()
- `float __magnitude` () const
- `void __merge` (`const __vector2list_info &__o`)
- `void __opr_find` (`std::size_t __size`)
- `void __opr_insert` (`std::size_t __pos`, `std::size_t __num`)
- `void __opr_iterate` (`std::size_t __num`)
- `void __resize` (`std::size_t __from`, `std::size_t`)
- `std::size_t __resize` ()
- `void __set_invalid` ()
- `void __set_list_cost` (`float __lc`)
- `void __set_vector_cost` (`float __vc`)
- `std::size_t __shift_count` ()
- `__stack_t __stack` () const
- `void __write` (`FILE *__f`) const

### Protected Attributes

- `__stack_t _M_stack`

#### 5.230.1 Detailed Description

A vector-to-list instrumentation line in the object table.

Definition at line 47 of file `profiler_vector_to_list.h`.

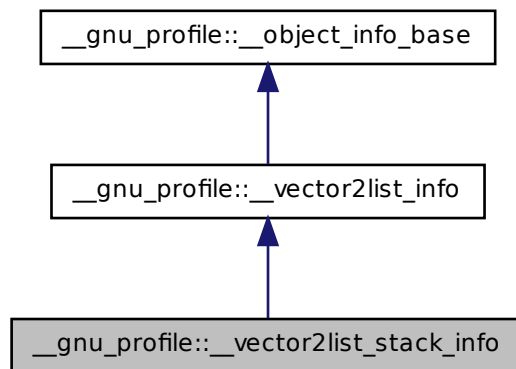
The documentation for this class was generated from the following file:

- [profiler\\_vector\\_to\\_list.h](#)

### 5.231 \_\_gnu\_profile::\_\_vector2list\_stack\_info Class Reference

A vector-to-list instrumentation line in the stack table.

Inheritance diagram for `__gnu_profile::__vector2list_stack_info`:



### Public Member Functions

- `__vector2list_stack_info` (const [\\_\\_vector2list\\_info](#) &\_\_o)
- [std::string](#) `__advice` () const
- `bool` `__is_valid` ()

- `bool __is_valid () const`
- `std::size_t __iterate ()`
- `float __list_cost ()`
- `float __magnitude () const`
- `void __merge (const __vector2list_info &__o)`
- `void __opr_find (std::size_t __size)`
- `void __opr_insert (std::size_t __pos, std::size_t __num)`
- `void __opr_iterate (std::size_t __num)`
- `void __resize (std::size_t __from, std::size_t)`
- `std::size_t __resize ()`
- `void __set_invalid ()`
- `void __set_list_cost (float __lc)`
- `void __set_vector_cost (float __vc)`
- `std::size_t __shift_count ()`
- `__stack_t __stack () const`
- `void __write (FILE *__f) const`

### Protected Attributes

- `__stack_t _M_stack`

#### 5.231.1 Detailed Description

A vector-to-list instrumentation line in the stack table.

Definition at line 155 of file `profiler_vector_to_list.h`.

The documentation for this class was generated from the following file:

- [profiler\\_vector\\_to\\_list.h](#)

### 5.232 `__gnu_profile::__warning_data` Struct Reference

Representation of a warning.

#### Public Member Functions

- `__warning_data (float __m, __stack_t __c, const char *__id, const std::string &__msg)`
- `bool operator< (const \_\_warning\_data &__other) const`

## 5.233 `std::__atomic0::__atomic_base< _ITp >` Struct Template Reference 1372

### Public Attributes

- `__stack_t __context`
- `float __magnitude`
- `const char * __warning_id`
- [std::string](#) `__warning_message`

### 5.232.1 Detailed Description

Representation of a warning.

Definition at line 80 of file `profiler_trace.h`.

The documentation for this struct was generated from the following file:

- [profiler\\_trace.h](#)

## 5.233 `std::__atomic0::__atomic_base< _ITp >` Struct Template Reference

Base class for atomic integrals.

### Public Member Functions

- `__atomic_base` (`const __atomic_base &`)
- `constexpr __atomic_base` (`__int_type __i`)
- `bool compare_exchange_strong` (`__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2`)
- `bool compare_exchange_strong` (`__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2`) `volatile`
- `bool compare_exchange_strong` (`__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst`)
- `bool compare_exchange_strong` (`__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst`) `volatile`
- `bool compare_exchange_weak` (`__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2`)
- `bool compare_exchange_weak` (`__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2`) `volatile`
- `bool compare_exchange_weak` (`__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst`)
- `bool compare_exchange_weak` (`__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst`) `volatile`
- `__int_type exchange` (`__int_type __i, memory_order __m=memory_order_seq_cst`) `volatile`

- `__int_type exchange (__int_type __i, memory_order __m=memory_order_seq_cst)`
- `__int_type fetch_add (__int_type __i, memory_order __m=memory_order_seq_cst)`
- `__int_type fetch_add (__int_type __i, memory_order __m=memory_order_seq_cst) volatile`
- `__int_type fetch_and (__int_type __i, memory_order __m=memory_order_seq_cst)`
- `__int_type fetch_and (__int_type __i, memory_order __m=memory_order_seq_cst) volatile`
- `__int_type fetch_or (__int_type __i, memory_order __m=memory_order_seq_cst)`
- `__int_type fetch_or (__int_type __i, memory_order __m=memory_order_seq_cst) volatile`
- `__int_type fetch_sub (__int_type __i, memory_order __m=memory_order_seq_cst)`
- `__int_type fetch_sub (__int_type __i, memory_order __m=memory_order_seq_cst) volatile`
- `__int_type fetch_xor (__int_type __i, memory_order __m=memory_order_seq_cst)`
- `__int_type fetch_xor (__int_type __i, memory_order __m=memory_order_seq_cst) volatile`
- `bool is_lock_free () const`
- `bool is_lock_free () const volatile`
- `__int_type load (memory_order __m=memory_order_seq_cst) const`
- `__int_type load (memory_order __m=memory_order_seq_cst) const volatile`
- `operator __int_type () const volatile`
- `operator __int_type () const`
- `__int_type operator&= (__int_type __i) volatile`
- `__int_type operator&= (__int_type __i)`
- `__int_type operator++ () volatile`
- `__int_type operator++ (int) volatile`
- `__int_type operator++ (int)`
- `__int_type operator++ ()`
- `__int_type operator+= (__int_type __i)`
- `__int_type operator+= (__int_type __i) volatile`
- `__int_type operator-- ()`
- `__int_type operator-- () volatile`
- `__int_type operator-- (int)`
- `__int_type operator-- (int) volatile`
- `__int_type operator-= (__int_type __i)`
- `__int_type operator-= (__int_type __i) volatile`
- `__int_type operator-= (__int_type __i) volatile`



- [\\_\\_atomic\\_base](#) & **operator=** (const [\\_\\_atomic\\_base](#) &)
- [\\_\\_atomic\\_base](#) & **operator=** (const [\\_\\_atomic\\_base](#) &) volatile
- [\\_\\_int\\_type](#) **operator=** ([\\_\\_int\\_type](#) \_\_i)
- [\\_\\_int\\_type](#) **operator^=** ([\\_\\_int\\_type](#) \_\_i) volatile
- [\\_\\_int\\_type](#) **operator^=** ([\\_\\_int\\_type](#) \_\_i)
- [\\_\\_int\\_type](#) **operator|=** ([\\_\\_int\\_type](#) \_\_i)
- [\\_\\_int\\_type](#) **operator|=** ([\\_\\_int\\_type](#) \_\_i) volatile
- void **store** ([\\_\\_int\\_type](#) \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst)
- void **store** ([\\_\\_int\\_type](#) \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile

### 5.233.1 Detailed Description

**template<typename \_ITp> struct std::\_\_atomic0::\_\_atomic\_base<\_ITp >**

Base class for atomic integrals.

Definition at line 455 of file atomic\_0.h.

The documentation for this struct was generated from the following file:

- [atomic\\_0.h](#)

## 5.234 std::\_\_atomic0::atomic\_address Struct Reference

[atomic\\_address](#)

### Public Member Functions

- **atomic\_address** (const [atomic\\_address](#) &)
- constexpr **atomic\_address** (void \*\_\_v)
- bool **compare\_exchange\_strong** (void \*&\_\_v1, void \*\_\_v2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2)
- bool **compare\_exchange\_strong** (void \*&\_\_v1, void \*\_\_v2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile
- bool **compare\_exchange\_strong** (void \*&\_\_v1, void \*\_\_v2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst)
- bool **compare\_exchange\_strong** (void \*&\_\_v1, void \*\_\_v2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile
- bool **compare\_exchange\_strong** (const void \*&\_\_v1, const void \*\_\_v2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2)
- bool **compare\_exchange\_strong** (const void \*&\_\_v1, const void \*\_\_v2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile

- bool **compare\_exchange\_strong** (const void \*&\_\_v1, const void \*\_\_v2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst)
- bool **compare\_exchange\_strong** (const void \*&\_\_v1, const void \*\_\_v2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile
- bool **compare\_exchange\_weak** (void \*&\_\_v1, void \*\_\_v2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile
- bool **compare\_exchange\_weak** (void \*&\_\_v1, void \*\_\_v2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst)
- bool **compare\_exchange\_weak** (void \*&\_\_v1, void \*\_\_v2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile
- bool **compare\_exchange\_weak** (const void \*&\_\_v1, const void \*\_\_v2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2)
- bool **compare\_exchange\_weak** (const void \*&\_\_v1, const void \*\_\_v2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst)
- bool **compare\_exchange\_weak** (const void \*&\_\_v1, const void \*\_\_v2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile
- bool **compare\_exchange\_weak** (const void \*&\_\_v1, const void \*\_\_v2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile
- bool **compare\_exchange\_weak** (void \*&\_\_v1, void \*\_\_v2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2)
- void \* **exchange** (void \*\_\_v, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst)
- void \* **exchange** (void \*\_\_v, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile
- void \* **fetch\_add** (ptrdiff\_t \_\_d, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile
- void \* **fetch\_add** (ptrdiff\_t \_\_d, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst)
- void \* **fetch\_sub** (ptrdiff\_t \_\_d, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst)
- void \* **fetch\_sub** (ptrdiff\_t \_\_d, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile
- bool **is\_lock\_free** () const volatile
- bool **is\_lock\_free** () const
- void \* **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const
- void \* **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const volatile
- **operator void \*** () const
- **operator void \*** () const volatile
- void \* **operator+=** (ptrdiff\_t \_\_d) volatile
- void \* **operator+=** (ptrdiff\_t \_\_d)
- void \* **operator-=** (ptrdiff\_t \_\_d) volatile
- void \* **operator-=** (ptrdiff\_t \_\_d)
- [atomic\\_address](#) & **operator=** (const [atomic\\_address](#) &) volatile
- void \* **operator=** (void \*\_\_v) volatile
- [atomic\\_address](#) & **operator=** (const [atomic\\_address](#) &)
- void \* **operator=** (void \*\_\_v)
- void **store** (void \*\_\_v, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile
- void **store** (void \*\_\_v, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst)

### 5.234.1 Detailed Description

[atomic\\_address](#)

Definition at line 138 of file atomic\_0.h.

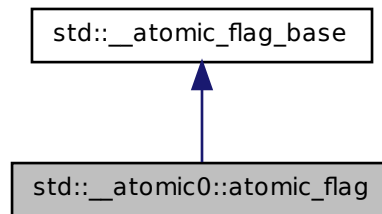
The documentation for this struct was generated from the following file:

- [atomic\\_0.h](#)

### 5.235 std::\_\_atomic0::atomic\_flag Struct Reference

[atomic\\_flag](#)

Inheritance diagram for std::\_\_atomic0::atomic\_flag:



#### Public Member Functions

- **atomic\_flag** (const [atomic\\_flag](#) &)
- **atomic\_flag** (bool \_\_i)
- void **clear** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst)
- void **clear** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile
- [atomic\\_flag](#) & **operator=** (const [atomic\\_flag](#) &) volatile
- [atomic\\_flag](#) & **operator=** (const [atomic\\_flag](#) &)
- bool **test\_and\_set** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile
- bool **test\_and\_set** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst)

#### Public Attributes

- bool **\_M\_i**

## 5.236 `std::__atomic2::__atomic_base<_ITp>` Struct Template Reference 1377

### 5.235.1 Detailed Description

[atomic\\_flag](#)

Definition at line 112 of file `atomic_0.h`.

The documentation for this struct was generated from the following file:

- [atomic\\_0.h](#)

## 5.236 `std::__atomic2::__atomic_base<_ITp>` Struct Template Reference

Base class for atomic integrals.

### Public Member Functions

- `__atomic_base` (const [\\_\\_atomic\\_base](#) &)
- constexpr `__atomic_base` (\_\_int\_type \_\_i)
- bool `compare_exchange_strong` (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2)
- bool `compare_exchange_strong` (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile
- bool `compare_exchange_strong` (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#))
- bool `compare_exchange_strong` (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) volatile
- bool `compare_exchange_weak` (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2)
- bool `compare_exchange_weak` (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile
- bool `compare_exchange_weak` (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#))
- bool `compare_exchange_weak` (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) volatile
- \_\_int\_type `exchange` (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) volatile
- \_\_int\_type `exchange` (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#))
- \_\_int\_type `fetch_add` (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#))
- \_\_int\_type `fetch_add` (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=[memory\\_order\\_seq\\_cst](#)) volatile

- `__int_type fetch_and (__int_type __i, memory_order __m=memory_order_seq_cst)`
- `__int_type fetch_and (__int_type __i, memory_order __m=memory_order_seq_cst) volatile`
- `__int_type fetch_or (__int_type __i, memory_order __m=memory_order_seq_cst)`
- `__int_type fetch_or (__int_type __i, memory_order __m=memory_order_seq_cst) volatile`
- `__int_type fetch_sub (__int_type __i, memory_order __m=memory_order_seq_cst)`
- `__int_type fetch_sub (__int_type __i, memory_order __m=memory_order_seq_cst) volatile`
- `__int_type fetch_xor (__int_type __i, memory_order __m=memory_order_seq_cst)`
- `__int_type fetch_xor (__int_type __i, memory_order __m=memory_order_seq_cst) volatile`
- `bool is_lock_free () const`
- `bool is_lock_free () const volatile`
- `__int_type load (memory_order __m=memory_order_seq_cst) const`
- `__int_type load (memory_order __m=memory_order_seq_cst) const volatile`
- `operator __int_type () const volatile`
- `operator __int_type () const`
- `__int_type operator&= (__int_type __i) volatile`
- `__int_type operator&= (__int_type __i)`
- `__int_type operator++ () volatile`
- `__int_type operator++ (int) volatile`
- `__int_type operator++ (int)`
- `__int_type operator++ ()`
- `__int_type operator+= (__int_type __i)`
- `__int_type operator+= (__int_type __i) volatile`
- `__int_type operator-- ()`
- `__int_type operator-- () volatile`
- `__int_type operator-- (int)`
- `__int_type operator-- (int) volatile`
- `__int_type operator-= (__int_type __i)`
- `__int_type operator-= (__int_type __i) volatile`
- `__int_type operator-= (__int_type __i) volatile`
- `__atomic_base & operator= (const __atomic_base &)`
- `__atomic_base & operator= (const __atomic_base &) volatile`
- `__int_type operator= (__int_type __i)`
- `__int_type operator^= (__int_type __i) volatile`
- `__int_type operator^= (__int_type __i)`
- `__int_type operator|= (__int_type __i)`

- `__int_type operator|= (__int_type __i) volatile`
- `void store (__int_type __i, memory\_order __m=memory_order_seq_cst)`
- `void store (__int_type __i, memory\_order __m=memory_order_seq_cst) volatile`

### 5.236.1 Detailed Description

`template<typename _ITp> struct std::__atomic2::__atomic_base<_ITp >`

Base class for atomic integrals.

Definition at line 439 of file `atomic_2.h`.

The documentation for this struct was generated from the following file:

- [atomic\\_2.h](#)

## 5.237 std::\_\_atomic2::atomic\_address Struct Reference

[atomic\\_address](#)

### Public Member Functions

- `atomic_address (const atomic\_address &)`
- `constexpr atomic_address (void *__v)`
- `bool compare_exchange_strong (void *&__v1, void *__v2, memory\_order __m1, memory\_order __m2)`
- `bool compare_exchange_strong (void *&__v1, void *__v2, memory\_order __m1, memory\_order __m2) volatile`
- `bool compare_exchange_strong (void *&__v1, void *__v2, memory\_order __m=memory_order_seq_cst)`
- `bool compare_exchange_strong (void *&__v1, void *__v2, memory\_order __m=memory_order_seq_cst) volatile`
- `bool compare_exchange_strong (const void *&__v1, const void *__v2, memory\_order __m1, memory\_order __m2)`
- `bool compare_exchange_strong (const void *&__v1, const void *__v2, memory\_order __m1, memory\_order __m2) volatile`
- `bool compare_exchange_strong (const void *&__v1, const void *__v2, memory\_order __m=memory_order_seq_cst)`
- `bool compare_exchange_strong (const void *&__v1, const void *__v2, memory\_order __m=memory_order_seq_cst) volatile`
- `bool compare_exchange_weak (void *&__v1, void *__v2, memory\_order __m1, memory\_order __m2) volatile`

- bool **compare\_exchange\_weak** (void \*&\_\_v1, void \*\_\_v2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst)
- bool **compare\_exchange\_weak** (void \*&\_\_v1, void \*\_\_v2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile
- bool **compare\_exchange\_weak** (const void \*&\_\_v1, const void \*\_\_v2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2)
- bool **compare\_exchange\_weak** (const void \*&\_\_v1, const void \*\_\_v2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst)
- bool **compare\_exchange\_weak** (const void \*&\_\_v1, const void \*\_\_v2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile
- bool **compare\_exchange\_weak** (const void \*&\_\_v1, const void \*\_\_v2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile
- bool **compare\_exchange\_weak** (void \*&\_\_v1, void \*\_\_v2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2)
- void \* **exchange** (void \*\_\_v, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst)
- void \* **exchange** (void \*\_\_v, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile
- void \* **fetch\_add** (ptrdiff\_t \_\_d, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile
- void \* **fetch\_add** (ptrdiff\_t \_\_d, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst)
- void \* **fetch\_sub** (ptrdiff\_t \_\_d, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst)
- void \* **fetch\_sub** (ptrdiff\_t \_\_d, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile
- bool **is\_lock\_free** () const volatile
- bool **is\_lock\_free** () const
- void \* **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const
- void \* **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const volatile
- **operator void \*** () const
- **operator void \*** () const volatile
- void \* **operator+=** (ptrdiff\_t \_\_d) volatile
- void \* **operator+=** (ptrdiff\_t \_\_d)
- void \* **operator-=** (ptrdiff\_t \_\_d) volatile
- void \* **operator-=** (ptrdiff\_t \_\_d)
- [atomic\\_address](#) & **operator=** (const [atomic\\_address](#) &) volatile
- void \* **operator=** (void \*\_\_v) volatile
- [atomic\\_address](#) & **operator=** (const [atomic\\_address](#) &)
- void \* **operator=** (void \*\_\_v)
- void **store** (void \*\_\_v, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile
- void **store** (void \*\_\_v, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst)

### 5.237.1 Detailed Description

[atomic\\_address](#)

Definition at line 105 of file atomic\_2.h.

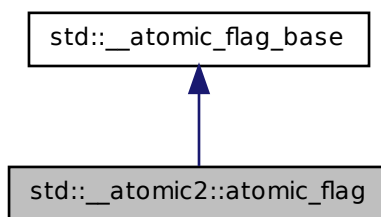
The documentation for this struct was generated from the following file:

- [atomic\\_2.h](#)

## 5.238 std::\_\_atomic2::atomic\_flag Struct Reference

[atomic\\_flag](#)

Inheritance diagram for std::\_\_atomic2::atomic\_flag:



### Public Member Functions

- **atomic\_flag** (bool \_\_i)
- **atomic\_flag** (const [atomic\\_flag](#) &)
- [atomic\\_flag](#) & **operator=** (const [atomic\\_flag](#) &) volatile
- [atomic\\_flag](#) & **operator=** (const [atomic\\_flag](#) &)

### Public Attributes

- bool **\_M\_i**



### 5.238.1 Detailed Description

[atomic\\_flag](#)

Definition at line 49 of file atomic\_2.h.

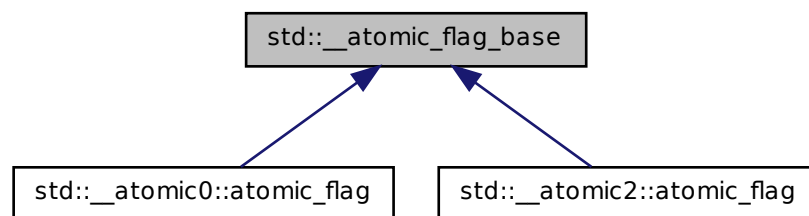
The documentation for this struct was generated from the following file:

- [atomic\\_2.h](#)

### 5.239 std::\_\_atomic\_flag\_base Struct Reference

Base type for atomic\_flag.

Inheritance diagram for std::\_\_atomic\_flag\_base:



#### Public Attributes

- `bool _M_i`

### 5.239.1 Detailed Description

Base type for atomic\_flag. Base type is POD with data, allowing atomic\_flag to derive from it and meet the standard layout type requirement. In addition to compatibility with a C interface, this allows different implementations of atomic\_flag to use the same atomic operation functions, via a standard conversion to the [\\_\\_atomic\\_flag\\_base](#) argument.

Definition at line 92 of file atomic\_base.h.

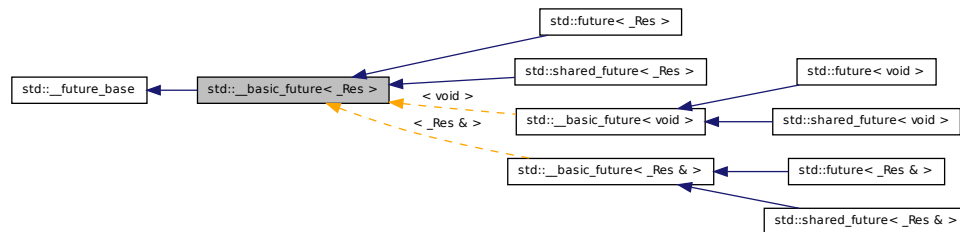
The documentation for this struct was generated from the following file:

- [atomic\\_base.h](#)

## 5.240 `std::__basic_future< _Res >` Class Template Reference

Common implementation for future and [shared\\_future](#).

Inheritance diagram for `std::__basic_future< _Res >`:



### Public Member Functions

- `__basic_future` (const `__basic_future` &)
- `__basic_future` & **operator=** (const `__basic_future` &)
- bool **valid** () const
- void **wait** () const
- template<typename \_Rep, typename \_Period >  
bool **wait\_for** (const `chrono::duration`< \_Rep, \_Period > &\_\_rel) const
- template<typename \_Clock, typename \_Duration >  
bool **wait\_until** (const `chrono::time_point`< \_Clock, \_Duration > &\_\_abs)  
const

### Static Public Member Functions

- template<typename \_Res, typename \_Allocator >  
static `_Ptr`< `_Result_alloc`< \_Res, \_Allocator > >::type **\_S\_allocate\_result**  
(const \_Allocator &\_\_a)

### Protected Types

- typedef `__future_base::_Result`< \_Res > & **\_\_result\_type**
- typedef `shared_ptr`< \_State > **\_\_state\_type**

### Protected Member Functions

- `__basic_future` (const `__state_type` &\_\_state)
- `__basic_future` (const `shared_future`< `_Res` > &)
- `__basic_future` (`shared_future`< `_Res` > &&)
- `__basic_future` (`future`< `_Res` > &&)
- `__result_type` `_M_get_result` ()
- `void` `_M_swap` (`__basic_future` &\_\_that)

#### 5.240.1 Detailed Description

`template<typename _Res> class std::__basic_future< _Res >`

Common implementation for future and `shared_future`.

Definition at line 495 of file future.

#### 5.240.2 Member Function Documentation

5.240.2.1 `template<typename _Res> __result_type std::__basic_future< _Res >::_M_get_result ( ) [inline, protected]`

Wait for the state to be ready and rethrow any stored exception.

Definition at line 538 of file future.

Referenced by `std::shared_future< _Res >::get()`, `std::future< void >::get()`, and `std::future< _Res >::get()`.

The documentation for this class was generated from the following file:

- `future`

### 5.241 `std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >` Class Template Reference

Common base for codecvt functions.

Inheritance diagram for `std::__codecvt_abstract_base< _InternT, _ExternT, _StateT`

```

graph TD
 subgraph State_Types [State Types]
 S1[std::codecv_state< char, char, mbstate_t >]
 S2[std::codecv_state< wchar_t, char, mbstate_t >]
 S3[std::codecv_state< int8_t, char, mbstate_t >]
 S4[std::codecv_state< int16_t, char, mbstate_t >]
 S5[std::codecv_state< int32_t, char, mbstate_t >]
 S6[std::codecv_state< int64_t, char, mbstate_t >]
 S7[std::codecv_state< float, char, mbstate_t >]
 S8[std::codecv_state< double, char, mbstate_t >]
 S9[std::codecv_state< int8_t, wchar_t, mbstate_t >]
 S10[std::codecv_state< int16_t, wchar_t, mbstate_t >]
 S11[std::codecv_state< int32_t, wchar_t, mbstate_t >]
 S12[std::codecv_state< int64_t, wchar_t, mbstate_t >]
 S13[std::codecv_state< float, wchar_t, mbstate_t >]
 S14[std::codecv_state< double, wchar_t, mbstate_t >]
 S15[std::codecv_state< int8_t, char, mbstate_t >]
 S16[std::codecv_state< int16_t, char, mbstate_t >]
 S17[std::codecv_state< int32_t, char, mbstate_t >]
 S18[std::codecv_state< int64_t, char, mbstate_t >]
 S19[std::codecv_state< float, char, mbstate_t >]
 S20[std::codecv_state< double, char, mbstate_t >]
 S21[std::codecv_state< int8_t, wchar_t, mbstate_t >]
 S22[std::codecv_state< int16_t, wchar_t, mbstate_t >]
 S23[std::codecv_state< int32_t, wchar_t, mbstate_t >]
 S24[std::codecv_state< int64_t, wchar_t, mbstate_t >]
 S25[std::codecv_state< float, wchar_t, mbstate_t >]
 S26[std::codecv_state< double, wchar_t, mbstate_t >]
 end

 subgraph Encoding_Types [Encoding Types]
 E1[std::codecv_encoding_state< char, char, mbstate_t >]
 E2[std::codecv_encoding_state< wchar_t, char, mbstate_t >]
 E3[std::codecv_encoding_state< int8_t, char, mbstate_t >]
 E4[std::codecv_encoding_state< int16_t, char, mbstate_t >]
 E5[std::codecv_encoding_state< int32_t, char, mbstate_t >]
 E6[std::codecv_encoding_state< int64_t, char, mbstate_t >]
 E7[std::codecv_encoding_state< float, char, mbstate_t >]
 E8[std::codecv_encoding_state< double, char, mbstate_t >]
 E9[std::codecv_encoding_state< int8_t, wchar_t, mbstate_t >]
 E10[std::codecv_encoding_state< int16_t, wchar_t, mbstate_t >]
 E11[std::codecv_encoding_state< int32_t, wchar_t, mbstate_t >]
 E12[std::codecv_encoding_state< int64_t, wchar_t, mbstate_t >]
 E13[std::codecv_encoding_state< float, wchar_t, mbstate_t >]
 E14[std::codecv_encoding_state< double, wchar_t, mbstate_t >]
 E15[std::codecv_encoding_state< int8_t, char, mbstate_t >]
 E16[std::codecv_encoding_state< int16_t, char, mbstate_t >]
 E17[std::codecv_encoding_state< int32_t, char, mbstate_t >]
 E18[std::codecv_encoding_state< int64_t, char, mbstate_t >]
 E19[std::codecv_encoding_state< float, char, mbstate_t >]
 E20[std::codecv_encoding_state< double, char, mbstate_t >]
 E21[std::codecv_encoding_state< int8_t, wchar_t, mbstate_t >]
 E22[std::codecv_encoding_state< int16_t, wchar_t, mbstate_t >]
 E23[std::codecv_encoding_state< int32_t, wchar_t, mbstate_t >]
 E24[std::codecv_encoding_state< int64_t, wchar_t, mbstate_t >]
 E25[std::codecv_encoding_state< float, wchar_t, mbstate_t >]
 E26[std::codecv_encoding_state< double, wchar_t, mbstate_t >]
 end

 subgraph Abstract_Base [Abstract Base]
 AB1[std::codecv_abstract_base< char, char, mbstate_t >]
 AB2[std::codecv_abstract_base< wchar_t, char, mbstate_t >]
 AB3[std::codecv_abstract_base< int8_t, char, mbstate_t >]
 AB4[std::codecv_abstract_base< int16_t, char, mbstate_t >]
 AB5[std::codecv_abstract_base< int32_t, char, mbstate_t >]
 AB6[std::codecv_abstract_base< int64_t, char, mbstate_t >]
 AB7[std::codecv_abstract_base< float, char, mbstate_t >]
 AB8[std::codecv_abstract_base< double, char, mbstate_t >]
 AB9[std::codecv_abstract_base< int8_t, wchar_t, mbstate_t >]
 AB10[std::codecv_abstract_base< int16_t, wchar_t, mbstate_t >]
 AB11[std::codecv_abstract_base< int32_t, wchar_t, mbstate_t >]
 AB12[std::codecv_abstract_base< int64_t, wchar_t, mbstate_t >]
 AB13[std::codecv_abstract_base< float, wchar_t, mbstate_t >]
 AB14[std::codecv_abstract_base< double, wchar_t, mbstate_t >]
 AB15[std::codecv_abstract_base< int8_t, char, mbstate_t >]
 AB16[std::codecv_abstract_base< int16_t, char, mbstate_t >]
 AB17[std::codecv_abstract_base< int32_t, char, mbstate_t >]
 AB18[std::codecv_abstract_base< int64_t, char, mbstate_t >]
 AB19[std::codecv_abstract_base< float, char, mbstate_t >]
 AB20[std::codecv_abstract_base< double, char, mbstate_t >]
 AB21[std::codecv_abstract_base< int8_t, wchar_t, mbstate_t >]
 AB22[std::codecv_abstract_base< int16_t, wchar_t, mbstate_t >]
 AB23[std::codecv_abstract_base< int32_t, wchar_t, mbstate_t >]
 AB24[std::codecv_abstract_base< int64_t, wchar_t, mbstate_t >]
 AB25[std::codecv_abstract_base< float, wchar_t, mbstate_t >]
 AB26[std::codecv_abstract_base< double, wchar_t, mbstate_t >]
 end

 subgraph Concrete_Base [Concrete Base]
 CB1[std::codecv_base]
 CB2[std::codecv_base]
 end

 subgraph Localize_Facet [Localize Facet]
 LF1[std::localize_facet]
 LF2[std::localize_facet]
 end

 S1 --> AB1
 S2 --> AB2
 S3 --> AB3
 S4 --> AB4
 S5 --> AB5
 S6 --> AB6
 S7 --> AB7
 S8 --> AB8
 S9 --> AB9
 S10 --> AB10
 S11 --> AB11
 S12 --> AB12
 S13 --> AB13
 S14 --> AB14
 S15 --> AB15
 S16 --> AB16
 S17 --> AB17
 S18 --> AB18
 S19 --> AB19
 S20 --> AB20
 S21 --> AB21
 S22 --> AB22
 S23 --> AB23
 S24 --> AB24
 S25 --> AB25
 S26 --> AB26

 AB1 --> CB1
 AB2 --> CB2
 AB3 --> CB1
 AB4 --> CB1
 AB5 --> CB1
 AB6 --> CB1
 AB7 --> CB1
 AB8 --> CB1
 AB9 --> CB2
 AB10 --> CB2
 AB11 --> CB2
 AB12 --> CB2
 AB13 --> CB2
 AB14 --> CB2
 AB15 --> CB1
 AB16 --> CB1
 AB17 --> CB1
 AB18 --> CB1
 AB19 --> CB1
 AB20 --> CB1
 AB21 --> CB2
 AB22 --> CB2
 AB23 --> CB2
 AB24 --> CB2
 AB25 --> CB2
 AB26 --> CB2

 CB1 --> LF1
 CB2 --> LF2

```

- typedef \_ExternT **extern\_type**
- typedef \_InternT **intern\_type**
- typedef codecvt\_base::result **result**
- typedef StateT **state\_type**

- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \* \_\_from, const extern\_type \* \_\_from\_end, const extern\_type \*&\_\_from\_next, intern\_type \* \_\_to, intern\_type \* \_\_to\_end, intern\_type \*&\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \* \_\_from, const extern\_type \* \_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \* \_\_from, const intern\_type \* \_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \* \_\_to, extern\_type \* \_\_to\_end, extern\_type \*&\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \* \_\_to, extern\_type \* \_\_to\_end, extern\_type \*&\_\_to\_next) const

- **\_\_codecv\_t\_abstract\_base** (size\_t \_\_refs=0)
- virtual bool **do\_always\_noconv** () const =0 throw ()
- virtual int **do\_encoding** () const =0 throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*&\_\_to\_next) const =0
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const =0
- virtual int **do\_max\_length** () const =0 throw ()

- virtual result `do_out` (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const =0
- virtual result `do_unshift` (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const =0

#### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale` (\_\_c\_locale &\_\_cloc) throw ()
- static void `_S_create_c_locale` (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void `_S_destroy_c_locale` (\_\_c\_locale &\_\_cloc)
- static `__c_locale _S_get_c_locale` ()
- static const char \* `_S_get_c_name` () throw ()
- static `__c_locale _S_lc_ctype_c_locale` (\_\_c\_locale \_\_cloc, const char \*\_\_s)

#### Friends

- class `locale::Impl`

#### 5.241.1 Detailed Description

`template<typename _InternT, typename _ExternT, typename _StateT> class std::__codecvt_abstract_base<_InternT, _ExternT, _StateT>`

Common base for codecvt functions. This template class provides implementations of the public functions that forward to the protected virtual functions.

This template also provides abstract stubs for the protected virtual functions.

Definition at line 69 of file `codecvt.h`.

#### 5.241.2 Member Function Documentation

**5.241.2.1** `template<typename _InternT, typename _ExternT, typename _StateT> virtual result std::__codecvt_abstract_base<_InternT, _ExternT, _StateT>::do_out ( state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type *& __from_next, extern_type * __to, extern_type * __to_end, extern_type *& __to_next ) const [protected, pure virtual]`

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

**See also**

[out](#) for more information.

Implemented in `std::codecvt<_InternT, _ExternT, _StateT >`, `std::codecvt<char, char, mbstate_t >`, `std::codecvt<wchar_t, char, mbstate_t >`, and `std::codecvt<_InternT, _ExternT, encoding_state >`.

Referenced by `std::__codecvt_abstract_base<_InternT, _ExternT, encoding_state >::out()`.

**5.241.2.2** `template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base<_InternT, _ExternT, _StateT>::in ( state_type & __state, const extern_type * __from, const extern_type * __from_end, const extern_type * & __from_next, intern_type * __to, intern_type * __to_end, intern_type * & __to_next ) const` `[inline]`

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

**Parameters**

*state* Persistent conversion state data.

*from* Start of input.  
*from\_end* End of input.  
*from\_next* Returns start of unconverted data.  
*to* Start of output buffer.  
*to\_end* End of output buffer.  
*to\_next* Returns start of unused output area.

### Returns

`codecvt_base::result`.

Definition at line 197 of file `codecvt.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::underflow()`.

**5.241.2.3** `template<typename _InternT, typename _ExternT, typename  
_StateT> result std::__codecvt_abstract_base<_InternT, _ExternT,  
_StateT>::out ( state_type & __state, const intern_type * __from,  
const intern_type * __from_end, const intern_type *& __from_next,  
extern_type * __to, extern_type * __to_end, extern_type *&  
__to_next ) const [inline]`

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

### Parameters

*state* Persistent conversion state data.

*from* Start of input.  
*from\_end* End of input.  
*from\_next* Returns start of unconverted data.  
*to* Start of output buffer.  
*to\_end* End of output buffer.  
*to\_next* Returns start of unused output area.

### Returns

`codecvt_base::result`.

Definition at line 117 of file `codecvt.h`.

**5.241.2.4** `template<typename _InternT, typename _ExternT, typename  
 _StateT> result std::__codecvt_abstract_base<_InternT, _ExternT,  
 _StateT >::unshift ( state_type & __state, extern_type * __to,  
 extern_type * __to_end, extern_type *& __to_next ) const  
 [inline]`

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

### Parameters

*state* Persistent conversion state data.  
*to* Start of output buffer.  
*to\_end* End of output buffer.  
*to\_next* Returns start of unused output area.



**Returns**

`codecvt_base::result`.

Definition at line 156 of file `codecvt.h`.

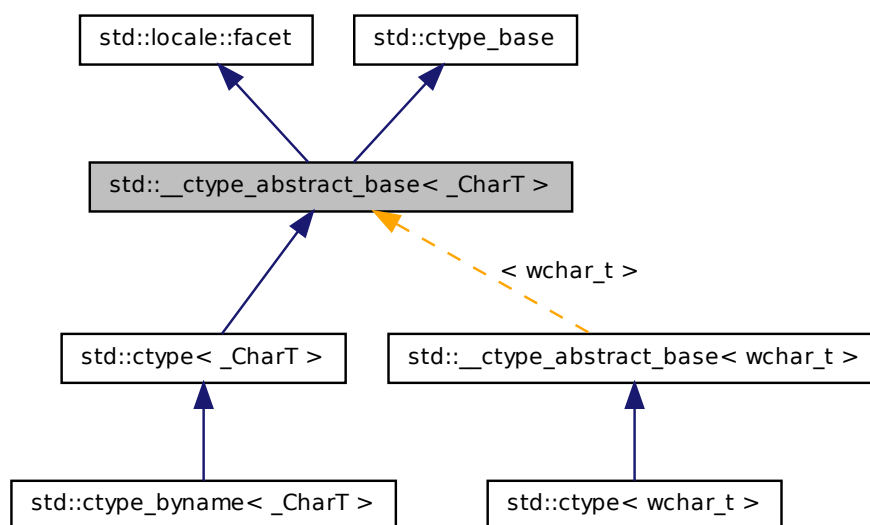
The documentation for this class was generated from the following file:

- [codecvt.h](#)

## 5.242 `std::__ctype_abstract_base< _CharT >` Class Template Reference

Common base for ctype facet.

Inheritance diagram for `std::__ctype_abstract_base< _CharT >`:

**Public Types**

- `typedef const int * __to_type`
- `typedef _CharT char_type`
- `typedef unsigned short mask`

### Public Member Functions

- `bool is (mask __m, char_type __c) const`
- `const char_type * is (const char_type *__lo, const char_type *__hi, mask *__vec) const`
- `char narrow (char_type __c, char __dfault) const`
- `const char_type * narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const`
- `const char_type * scan_is (mask __m, const char_type *__lo, const char_type *__hi) const`
- `const char_type * scan_not (mask __m, const char_type *__lo, const char_type *__hi) const`
- `const char_type * tolower (char_type *__lo, const char_type *__hi) const`
- `char_type tolower (char_type __c) const`
- `const char_type * toupper (char_type *__lo, const char_type *__hi) const`
- `char_type toupper (char_type __c) const`
- `const char * widen (const char *__lo, const char *__hi, char_type *__to) const`
- `char_type widen (char __c) const`

### Static Public Attributes

- `static const mask alnum`
- `static const mask alpha`
- `static const mask cntrl`
- `static const mask digit`
- `static const mask graph`
- `static const mask lower`
- `static const mask print`
- `static const mask punct`
- `static const mask space`
- `static const mask upper`
- `static const mask xdigit`

### Protected Member Functions

- `__ctype_abstract_base (size_t __refs=0)`
- `virtual const char_type * do_is (const char_type *__lo, const char_type *__hi, mask *__vec) const =0`
- `virtual bool do_is (mask __m, char_type __c) const =0`
- `virtual char do_narrow (char_type __c, char __dfault) const =0`
- `virtual const char_type * do_narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__dest) const =0`

- virtual const `char_type` \* `do_scan_is` (mask `__m`, const `char_type` \* `__lo`, const `char_type` \* `__hi`) const =0
- virtual const `char_type` \* `do_scan_not` (mask `__m`, const `char_type` \* `__lo`, const `char_type` \* `__hi`) const =0
- virtual `char_type` `do_tolower` (`char_type`) const =0
- virtual const `char_type` \* `do_tolower` (`char_type` \* `__lo`, const `char_type` \* `__hi`) const =0
- virtual `char_type` `do_toupper` (`char_type`) const =0
- virtual const `char_type` \* `do_toupper` (`char_type` \* `__lo`, const `char_type` \* `__hi`) const =0
- virtual const char \* `do_widen` (const char \* `__lo`, const char \* `__hi`, `char_type` \* `__dest`) const =0
- virtual `char_type` `do_widen` (char) const =0

#### Static Protected Member Functions

- static `__c_locale` `_S_clone_c_locale` (`__c_locale` & `__cloc`) throw ()
- static void `_S_create_c_locale` (`__c_locale` & `__cloc`, const char \* `__s`, `__c_locale` `__old`=0)
- static void `_S_destroy_c_locale` (`__c_locale` & `__cloc`)
- static `__c_locale` `_S_get_c_locale` ()
- static const char \* `_S_get_c_name` () throw ()
- static `__c_locale` `_S_lc_ctype_c_locale` (`__c_locale` `__cloc`, const char \* `__s`)

#### Friends

- class `locale::__Impl`

#### 5.242.1 Detailed Description

`template<typename _CharT> class std::__ctype_abstract_base<_CharT>`

Common base for ctype facet. This template class provides implementations of the public functions that forward to the protected virtual functions.

This template also provides abstract stubs for the protected virtual functions.

Definition at line 145 of file `locale_facets.h`.

## 5.242.2 Member Typedef Documentation

**5.242.2.1** `template<typename _CharT> typedef _CharT  
std::__ctype_abstract_base< _CharT >::char_type`

Typedef for the template parameter.

Reimplemented in [std::ctype< \\_CharT >](#), and [std::ctype< wchar\\_t >](#).

Definition at line 150 of file locale\_facets.h.

## 5.242.3 Member Function Documentation

**5.242.3.1** `template<typename _CharT> virtual bool std::__ctype_abstract_  
base< _CharT >::do_is ( mask __m, char_type __c ) const  
[protected, pure virtual]`

Test char\_type classification.

This function finds a mask M for *c* and compares it to mask *m*.

[do\\_is\(\)](#) is a hook for a derived facet to change the behavior of classifying. [do\\_is\(\)](#) must always return the same result for the same input.

### Parameters

*c* The char\_type to find the mask of.

*m* The mask to compare against.

### Returns

(M & m) != 0.

Implemented in [std::ctype< \\_CharT >](#), and [std::ctype< wchar\\_t >](#).

Referenced by [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >::is\(\)](#).

**5.242.3.2** `template<typename _CharT> virtual const char_type*  
std::__ctype_abstract_base< _CharT >::do_is ( const char_type *  
__lo, const char_type * __hi, mask * __vec ) const [protected,  
pure virtual]`

Return a mask array.

This function finds the mask for each `char_type` in the range `[lo,hi)` and successively writes it to `vec`. `vec` must have as many elements as the input.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

#### Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- vec* Pointer to an array of mask storage.

#### Returns

*hi*.

Implemented in `std::ctype<_CharT>`, and `std::ctype<wchar_t>`.

#### 5.242.3.3 `template<typename _CharT> virtual char std::__ctype_abstract_base<_CharT>::do_narrow( char_type, char __dfault ) const [protected, pure virtual]`

Narrow `char_type` to `char`.

This virtual function converts the argument to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

- c* The `char_type` to convert.
- dfault* Char to return if conversion fails.

#### Returns

The converted `char`.

Implemented in `std::ctype<_CharT>`, and `std::ctype<wchar_t>`.

Referenced by `std::__ctype_abstract_base<wchar_t>::narrow()`.

**5.242.3.4** `template<typename _CharT> virtual const char_type*  
 std::__ctype_abstract_base<_CharT>::do_narrow ( const  
 char_type * __lo, const char_type * __hi, char __dfault, char *  
 __dest ) const [protected, pure virtual]`

Narrow char\_type array to char.

This virtual function converts each char\_type in the range [lo,hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, *dfault* is used instead.

[do\\_narrow\(\)](#) is a hook for a derived facet to change the behavior of narrowing. [do\\_narrow\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- dfault* Char to use if conversion fails.
- to* Pointer to the destination array.

#### Returns

*hi*.

Implemented in [std::ctype<\\_CharT>](#), and [std::ctype<wchar\\_t>](#).

**5.242.3.5** `template<typename _CharT> virtual const char_type*  
 std::__ctype_abstract_base<_CharT>::do_scan_is ( mask  
 __m, const char_type * __lo, const char_type * __hi ) const  
 [protected, pure virtual]`

Find char\_type matching mask.

This function searches for and returns the first char\_type c in [lo,hi) for which `is(m,c)` is true.

[do\\_scan\\_is\(\)](#) is a hook for a derived facet to change the behavior of match searching. [do\\_is\(\)](#) must always return the same result for the same input.

#### Parameters

- m* The mask to compare against.

*lo* Pointer to start of range.

*hi* Pointer to end of range.

### Returns

Pointer to a matching `char_type` if found, else *hi*.

Implemented in `std::ctype<_CharT>`, and `std::ctype<wchar_t>`.

Referenced by `std::__ctype_abstract_base<wchar_t>::scan_is()`.

**5.242.3.6** `template<typename _CharT> virtual const char_type*  
std::__ctype_abstract_base<_CharT>::do_scan_not ( mask  
__m, const char_type * __lo, const char_type * __hi ) const  
[protected, pure virtual]`

Find `char_type` not matching mask.

This function searches for and returns a pointer to the first `char_type` *c* of [*lo*,*hi*) for which `is(m,c)` is false.

`do_scan_is()` is a hook for a derived facet to change the behavior of match searching.

`do_is()` must always return the same result for the same input.

### Parameters

*m* The mask to compare against.

*lo* Pointer to start of range.

*hi* Pointer to end of range.

### Returns

Pointer to a non-matching `char_type` if found, else *hi*.

Implemented in `std::ctype<_CharT>`, and `std::ctype<wchar_t>`.

Referenced by `std::__ctype_abstract_base<wchar_t>::scan_not()`.

**5.242.3.7** `template<typename _CharT> virtual char_type  
std::__ctype_abstract_base<_CharT>::do_tolower ( char_type )  
const [protected, pure virtual]`

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

#### Parameters

*c* The `char_type` to convert.

#### Returns

The lowercase `char_type` if convertible, else *c*.

Implemented in `std::ctype<_CharT>`, and `std::ctype<wchar_t>`.

Referenced by `std::__ctype_abstract_base<wchar_t>::tolower()`.

**5.242.3.8** `template<typename _CharT> virtual const char_type*  
std::__ctype_abstract_base<_CharT>::do_tolower ( char_type  
* __lo, const char_type * __hi ) const [protected, pure  
virtual]`

Convert array to lowercase.

This virtual function converts each `char_type` in the range `[lo,hi)` to lowercase if possible. Other elements remain untouched.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

#### Parameters

*lo* Pointer to start of range.

*hi* Pointer to end of range.

#### Returns

*hi*.

Implemented in `std::ctype<_CharT>`, and `std::ctype<wchar_t>`.

**5.242.3.9** `template<typename _CharT> virtual char_type  
std::__ctype_abstract_base<_CharT>::do_toupper ( char_type )  
const [protected, pure virtual]`



Convert to uppercase.

This virtual function converts the `char_type` argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

#### Parameters

*c* The `char_type` to convert.

#### Returns

The uppercase `char_type` if convertible, else *c*.

Implemented in `std::ctype<_CharT>`, and `std::ctype<wchar_t>`.

Referenced by `std::__ctype_abstract_base<wchar_t>::toupper()`.

**5.242.3.10** `template<typename _CharT> virtual const char_type*  
std::__ctype_abstract_base<_CharT>::do_toupper ( char_type  
* __lo, const char_type * __hi ) const [protected, pure  
virtual]`

Convert array to uppercase.

This virtual function converts each `char_type` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

#### Parameters

*lo* Pointer to start of range.

*hi* Pointer to end of range.

#### Returns

*hi*.

Implemented in `std::ctype<_CharT>`, and `std::ctype<wchar_t>`.

**5.242.3.11** `template<typename _CharT> virtual char_type  
std::__ctype_abstract_base<_CharT>::do_widen ( char ) const  
[protected, pure virtual]`

Widen char.

This virtual function converts the char to char\_type using the simplest reasonable transformation.

`do_widen()` is a hook for a derived facet to change the behavior of widening. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

*c* The char to convert.

#### Returns

The converted char\_type

Implemented in `std::ctype<_CharT>`, and `std::ctype<wchar_t>`.

Referenced by `std::__ctype_abstract_base<wchar_t>::widen()`.

**5.242.3.12** `template<typename _CharT> virtual const char*  
std::__ctype_abstract_base<_CharT>::do_widen ( const  
char * __lo, const char * __hi, char_type * __dest ) const  
[protected, pure virtual]`

Widen char array.

This function converts each char in the input to char\_type using the simplest reasonable transformation.

`do_widen()` is a hook for a derived facet to change the behavior of widening. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

*lo* Pointer to start range.

*hi* Pointer to end of range.

*to* Pointer to the destination array.

**Returns***hi*.Implemented in `std::ctype<_CharT>`, and `std::ctype<wchar_t>`.

**5.242.3.13** `template<typename _CharT> const char_type*`  
`std::__ctype_abstract_base<_CharT>::is ( const char_type *`  
`__lo, const char_type * __hi, mask * __vec ) const [inline]`

Return a mask array.

This function finds the mask for each `char_type` in the range `[lo,hi)` and successively writes it to `vec`. `vec` must have as many elements as the char array. It does so by returning the value of `ctype<char_type>::do_is()`.

**Parameters**

*lo* Pointer to start of range.  
*hi* Pointer to end of range.  
*vec* Pointer to an array of mask storage.

**Returns***hi*.Definition at line 180 of file `locale_facets.h`.

**5.242.3.14** `template<typename _CharT> bool std::__ctype_abstract_base<`  
`_CharT>::is ( mask __m, char_type __c ) const [inline]`

Test `char_type` classification.

This function finds a mask `M` for `c` and compares it to mask `m`. It does so by returning the value of `ctype<char_type>::do_is()`.

**Parameters**

*c* The `char_type` to compare the mask of.  
*m* The mask to compare against.

**Returns**`(M & m) != 0`.

## 5.242 std::\_\_ctype\_abstract\_base<\_CharT> Class Template Reference 1401

Definition at line 163 of file locale\_facets.h.

Referenced by std::regex\_traits<\_Ch\_type>::isctype(), and std::basic\_istream<\_CharT, \_Traits>::sentry::sentry().

**5.242.3.15** `template<typename _CharT> const char_type*  
std::__ctype_abstract_base<_CharT>::narrow ( const char_type  
* __lo, const char_type * __hi, char __dfault, char * __to ) const  
[inline]`

Narrow array to char array.

This function converts each `char_type` in the input to `char` using the simplest reasonable transformation and writes the results to the destination array. For any `char_type` in the input that cannot be converted, *dfault* is used instead. It does so by returning `ctype<char_type>::do_narrow(lo, hi, dfault, to)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

### Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- dfault* Char to use if conversion fails.
- to* Pointer to the destination array.

### Returns

*hi*.

Definition at line 347 of file locale\_facets.h.

**5.242.3.16** `template<typename _CharT> char std::__ctype_abstract_base<  
_CharT>::narrow ( char_type __c, char __dfault ) const  
[inline]`

Narrow `char_type` to `char`.

This function converts the `char_type` to `char` using the simplest reasonable transformation. If the conversion fails, *dfault* is returned instead. It does so by returning `ctype<char_type>::do_narrow(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

- c* The char\_type to convert.
- dfault* Char to return if conversion fails.

**Returns**

The converted char.

Definition at line 325 of file locale\_facets.h.

Referenced by std::time\_put< \_CharT, \_OutIter >::put().

**5.242.3.17** `template<typename _CharT> const char_type*  
std::__ctype_abstract_base< _CharT >::scan_is ( mask __m,  
const char_type * __lo, const char_type * __hi ) const [inline]`

Find char\_type matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is true. It does so by returning [ctype<char\\_type>::do\\_scan\\_is\(\)](#).

**Parameters**

- m* The mask to compare against.
- lo* Pointer to start of range.
- hi* Pointer to end of range.

**Returns**

Pointer to matching char\_type if found, else *hi*.

Definition at line 196 of file locale\_facets.h.

**5.242.3.18** `template<typename _CharT> const char_type*  
std::__ctype_abstract_base< _CharT >::scan_not ( mask __m,  
const char_type * __lo, const char_type * __hi ) const [inline]`

Find char\_type not matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is false. It does so by returning [ctype<char\\_type>::do\\_scan\\_not\(\)](#).

**Parameters**

- m* The mask to compare against.  
*lo* Pointer to first char in range.  
*hi* Pointer to end of range.

**Returns**

Pointer to non-matching char if found, else *hi*.

Definition at line 212 of file locale\_facets.h.

**5.242.3.19** `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::tolower ( char_type __c ) const [inline]`

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char\_type>::do\_toupper(c).

**Parameters**

- c* The char\_type to convert.

**Returns**

The lowercase char\_type if convertible, else *c*.

Definition at line 255 of file locale\_facets.h.

**5.242.3.20** `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::tolower ( char_type * __lo, const char_type * __hi ) const [inline]`

Convert array to lowercase.

This function converts each char\_type in the range [lo,hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char\_type>:: do\_toupper(lo, hi).

**Parameters**

- lo* Pointer to start of range.

*hi* Pointer to end of range.

#### Returns

*hi*.

Definition at line 270 of file locale\_facets.h.

**5.242.3.21** `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::toupper ( char_type __c ) const [inline]`

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_toupper()`.

#### Parameters

*c* The char\_type to convert.

#### Returns

The uppercase char\_type if convertible, else *c*.

Definition at line 226 of file locale\_facets.h.

**5.242.3.22** `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::toupper ( char_type * __lo, const char_type * __hi ) const [inline]`

Convert array to uppercase.

This function converts each char\_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

#### Parameters

*lo* Pointer to start of range.

*hi* Pointer to end of range.

**Returns**

*hi.*

Definition at line 241 of file locale\_facets.h.

**5.242.3.23** `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::widen ( char __c ) const [inline]`

Widen char to char\_type.

This function converts the char argument to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

*c* The char to convert.

**Returns**

The converted char\_type.

Definition at line 287 of file locale\_facets.h.

Referenced by std::money\_get<\_CharT, \_InIter>::do\_get(), std::time\_put<\_CharT, \_OutIter>::do\_put(), std::money\_put<\_CharT, \_OutIter>::do\_put(), std::regex\_traits<\_Ch\_type>::isctype(), and std::operator<<().

**5.242.3.24** `template<typename _CharT> const char* std::__ctype_abstract_base<_CharT>::widen ( const char * __lo, const char * __hi, char_type * __to ) const [inline]`

Widen array to char\_type.

This function converts each char in the input to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

*lo* Pointer to start of range.



*hi* Pointer to end of range.

*to* Pointer to the destination array.

### Returns

*hi*.

Definition at line 306 of file `locale_facets.h`.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

## 5.243 `std::__debug::bitset<_Nb>` Class Template Reference

Class [std::bitset](#) with additional safety/checking/debug instrumentation.

Inherits `bitset<_Nb>`.

### Public Types

- typedef `_Base::reference` **reference**

### Public Member Functions

- constexpr **bitset** (unsigned long long \_\_val)
- template<class \_CharT, class \_Traits, class \_Alloc >  
**bitset** (const [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_str, type-  
name [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc >::size\_type \_\_pos, typename  
[std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc >::size\_type \_\_n, \_CharT \_\_zero,  
\_CharT \_\_one=\_CharT('1'))
- **bitset** (const [\\_Base](#) &\_\_x)
- template<typename \_CharT, typename \_Traits, typename \_Alloc >  
**bitset** (const [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_str, type-  
name [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc >::size\_type \_\_pos=0, type-  
name [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc >::size\_type \_\_n=([std::basic\\_-](#)  
[string](#)< \_CharT, \_Traits, \_Alloc >::npos))
- template<typename \_CharT >  
**bitset** (const \_CharT \*\_\_str, typename [std::basic\\_string](#)< \_CharT >::size\_type  
\_\_n=[std::basic\\_string](#)< \_CharT >::npos, \_CharT \_\_zero=\_CharT('0'), \_CharT  
\_\_one=\_CharT('1'))
- [\\_Base](#) & [\\_M\\_base](#) ()
- const [\\_Base](#) & [\\_M\\_base](#) () const
- [bitset](#)<\_Nb> & **flip** ()

- [bitset](#)<\_Nb> & **flip** (size\_t \_\_pos)
- bool **operator!=** (const [bitset](#)<\_Nb> &\_\_rhs) const
- [bitset](#)<\_Nb> & **operator&=** (const [bitset](#)<\_Nb> &\_\_rhs)
- [bitset](#)<\_Nb> **operator<<** (size\_t \_\_pos) const
- [bitset](#)<\_Nb> & **operator<<=** (size\_t \_\_pos)
- bool **operator==** (const [bitset](#)<\_Nb> &\_\_rhs) const
- [bitset](#)<\_Nb> **operator>>** (size\_t \_\_pos) const
- [bitset](#)<\_Nb> & **operator>>=** (size\_t \_\_pos)
- reference **operator[ ]** (size\_t \_\_pos)
- bool **operator[ ]** (size\_t \_\_pos) const
- [bitset](#)<\_Nb> & **operator^=** (const [bitset](#)<\_Nb> &\_\_rhs)
- [bitset](#)<\_Nb> & **operator|=** (const [bitset](#)<\_Nb> &\_\_rhs)
- [bitset](#)<\_Nb> **operator~** () const
- [bitset](#)<\_Nb> & **reset** ()
- [bitset](#)<\_Nb> & **reset** (size\_t \_\_pos)
- [bitset](#)<\_Nb> & **set** (size\_t \_\_pos, bool \_\_val=true)
- [bitset](#)<\_Nb> & **set** ()
- [std::basic\\_string](#)< char, [std::char\\_traits](#)< char >, [std::allocator](#)< char > > **to\_string** (char \_\_zero, char \_\_one= '1') const
- template<typename \_CharT, typename \_Traits >  
[std::basic\\_string](#)< \_CharT, \_Traits, [std::allocator](#)< \_CharT > > **to\_string** () const
- template<typename \_CharT, typename \_Traits, typename \_Alloc >  
[std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > **to\_string** () const
- template<class \_CharT, class \_Traits >  
[std::basic\\_string](#)< \_CharT, \_Traits, [std::allocator](#)< \_CharT > > **to\_string** (\_CharT \_\_zero, \_CharT \_\_one=\_CharT('1')) const
- [std::basic\\_string](#)< char, [std::char\\_traits](#)< char >, [std::allocator](#)< char > > **to\_string** () const
- template<typename \_CharT >  
[std::basic\\_string](#)< \_CharT, [std::char\\_traits](#)< \_CharT >, [std::allocator](#)< \_CharT > > **to\_string** () const
- template<class \_CharT >  
[std::basic\\_string](#)< \_CharT, [std::char\\_traits](#)< \_CharT >, [std::allocator](#)< \_CharT > > **to\_string** (\_CharT \_\_zero, \_CharT \_\_one=\_CharT('1')) const
- template<class \_CharT, class \_Traits, class \_Alloc >  
[std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > **to\_string** (\_CharT \_\_zero, \_CharT \_\_one=\_CharT('1')) const

### 5.243.1 Detailed Description

template<size\_t \_Nb> class std::\_\_debug::bitset<\_Nb>

Class [std::bitset](#) with additional safety/checking/debug instrumentation.

## 5.244 `std::__debug::deque< _Tp, _Allocator >` Class Template Reference 1408

Definition at line 43 of file `debug/bitset`.

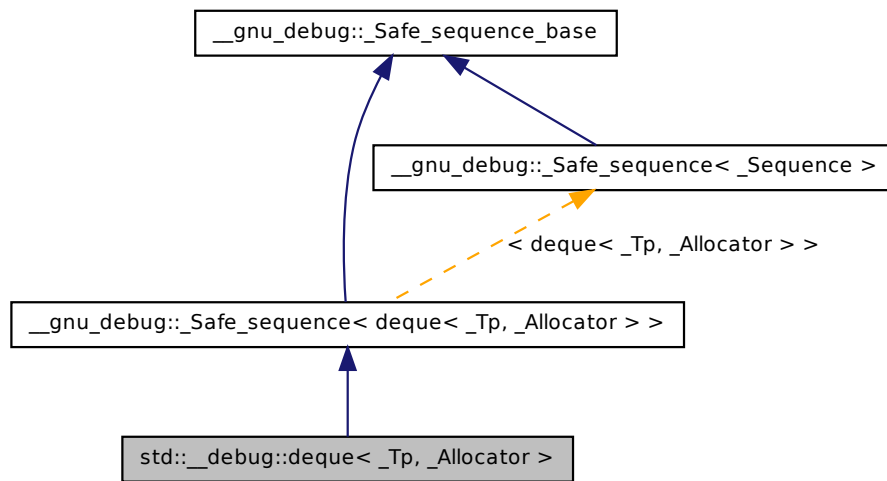
The documentation for this class was generated from the following file:

- [debug/bitset](#)

### 5.244 `std::__debug::deque< _Tp, _Allocator >` Class Template Reference

Class [std::deque](#) with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::deque< _Tp, _Allocator >`:



#### Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::__Safe_iterator< _Base_const_iterator, deque >` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**

- typedef `__gnu_debug::_Safe_iterator<_Base_iterator, deque >` `iterator`
- typedef `_Base::pointer` `pointer`
- typedef `_Base::reference` `reference`
- typedef `std::reverse_iterator< iterator >` `reverse_iterator`
- typedef `_Base::size_type` `size_type`
- typedef `_Tp` `value_type`

### Public Member Functions

- `deque` (const `_Allocator` &\_\_a=\_Allocator())
- `deque` (size\_type \_\_n)
- template<class `_InputIterator` >  
  `deque` (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Allocator` &\_\_a=\_Allocator())
- `deque` (`initializer_list`< value\_type > \_\_l, const allocator\_type &\_\_a=allocator\_type())
- `deque` (const `deque` &\_\_x)
- `deque` (size\_type \_\_n, const `_Tp` &\_\_value, const `_Allocator` &\_\_a=\_Allocator())
- `deque` (const `_Base` &\_\_x)
- `deque` (`deque` &&\_\_x)
- void `_M_attach` (`_Safe_iterator_base` \*\_\_it, bool \_\_constant)
- void `_M_attach_single` (`_Safe_iterator_base` \*\_\_it, bool \_\_constant) throw ()
- `_Base` & `_M_base` ()
- const `_Base` & `_M_base` () const
- void `_M_detach` (`_Safe_iterator_base` \*\_\_it)
- void `_M_detach_single` (`_Safe_iterator_base` \*\_\_it) throw ()
- void `_M_invalidate_all` () const
- void `_M_invalidate_if` (`_Predicate` \_\_pred)
- void `_M_transfer_from_if` (`_Safe_sequence` &\_\_from, `_Predicate` \_\_pred)
- void `assign` (`initializer_list`< value\_type > \_\_l)
- template<class `_InputIterator` >  
  void `assign` (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- void `assign` (size\_type \_\_n, const `_Tp` &\_\_t)
- reference `back` ()
- const\_reference `back` () const
- `iterator` `begin` ()
- `const_iterator` `begin` () const
- `const_iterator` `cbegin` () const
- `const_iterator` `cend` () const
- void `clear` ()
- `const_reverse_iterator` `crbegin` () const
- `const_reverse_iterator` `crend` () const

- `template<typename... _Args>`  
`iterator` **emplace** (`iterator` \_\_position, `_Args` &&...\_\_args)
- `template<typename... _Args>`  
`void` **emplace\_back** (`_Args` &&...\_\_args)
- `template<typename... _Args>`  
`void` **emplace\_front** (`_Args` &&...\_\_args)
- `iterator` **end** ()
- `const_iterator` **end** () const
- `iterator` **erase** (`iterator` \_\_position)
- `iterator` **erase** (`iterator` \_\_first, `iterator` \_\_last)
- reference **front** ()
- `const_reference` **front** () const
- `template<class _InputIterator >`  
`void` **insert** (`iterator` \_\_position, `_InputIterator` \_\_first, `_InputIterator` \_\_last)
- `void` **insert** (`iterator` \_\_position, `size_type` \_\_n, `const _Tp` &\_\_x)
- `iterator` **insert** (`iterator` \_\_position, `const _Tp` &\_\_x)
- `iterator` **insert** (`iterator` \_\_position, `_Tp` &&\_\_x)
- `void` **insert** (`iterator` \_\_p, `initializer_list`< `value_type` > \_\_l)
- `deque` & **operator=** (`const deque` &\_\_x)
- `deque` & **operator=** (`deque` &&\_\_x)
- `deque` & **operator=** (`initializer_list`< `value_type` > \_\_l)
- reference **operator[]** (`size_type` \_\_n)
- `const_reference` **operator[]** (`size_type` \_\_n) const
- `void` **pop\_back** ()
- `void` **pop\_front** ()
- `void` **push\_back** (`const _Tp` &\_\_x)
- `void` **push\_back** (`_Tp` &&\_\_x)
- `void` **push\_front** (`_Tp` &&\_\_x)
- `void` **push\_front** (`const _Tp` &\_\_x)
- `reverse_iterator` **rbegin** ()
- `const_reverse_iterator` **rbegin** () const
- `const_reverse_iterator` **rend** () const
- `reverse_iterator` **rend** ()
- `void` **resize** (`size_type` \_\_sz)
- `void` **resize** (`size_type` \_\_sz, `const _Tp` &\_\_c)
- `void` **swap** (`deque` &\_\_x)

#### Public Attributes

- `_Safe_iterator_base` \* `_M_const_iterators`
- `_Safe_iterator_base` \* `_M_iterators`
- unsigned int `_M_version`

## 5.244 std::\_\_debug::deque<\_Tp, \_Allocator > Class Template Reference 1411

---

### Protected Member Functions

- void [\\_M\\_detach\\_all](#) ()
- void [\\_M\\_detach\\_singular](#) ()
- [\\_\\_gnu\\_cxx::\\_\\_mutex](#) & [\\_M\\_get\\_mutex](#) () throw ()
- void [\\_M\\_revalidate\\_singular](#) ()
- void [\\_M\\_swap](#) (\_Safe\_sequence\_base &\_\_x)

### 5.244.1 Detailed Description

template<typename \_Tp, typename \_Allocator = std::allocator<\_Tp>> class  
std::\_\_debug::deque<\_Tp, \_Allocator >

Class [std::deque](#) with safety/checking/debug instrumentation.

Definition at line 43 of file debug/deque.

### 5.244.2 Member Function Documentation

5.244.2.1 void [\\_\\_gnu\\_debug::Safe\\_sequence\\_base::M\\_attach](#) (  
\_Safe\_iterator\_base \* *\_\_it*, bool *\_\_constant* ) [*inherited*]

Attach an iterator to this sequence.

5.244.2.2 void [\\_\\_gnu\\_debug::Safe\\_sequence\\_base::M\\_attach\\_single](#)  
( \_Safe\_iterator\_base \* *\_\_it*, bool *\_\_constant* ) throw ()  
[*inherited*]

Likewise but not thread safe.

5.244.2.3 void [\\_\\_gnu\\_debug::Safe\\_sequence\\_base::M\\_detach](#) (  
\_Safe\_iterator\_base \* *\_\_it* ) [*inherited*]

Detach an iterator from this sequence

5.244.2.4 void [\\_\\_gnu\\_debug::Safe\\_sequence\\_base::M\\_detach\\_all](#) ( )  
[*protected*, *inherited*]

Detach all iterators, leaving them singular.

Referenced by [\\_\\_gnu\\_debug::Safe\\_sequence\\_base::~~Safe\\_sequence\\_base](#)().

## **5.244 std::\_\_debug::deque<\_Tp, \_Allocator > Class Template Reference 1412**

**5.244.2.5** void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach\_single (   
\_Safe\_iterator\_base \* \_\_it ) throw () [inherited]

Likewise but not thread safe.

**5.244.2.6** void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach\_singular ( )   
[protected, inherited]

Detach all singular iterators.

### **Postcondition**

for all iterators i attached to this sequence, i->\_M\_version == \_M\_version.

**5.244.2.7** \_\_gnu\_cxx::mutex& \_\_gnu\_debug::Safe\_sequence\_  
base::M\_get\_mutex ( ) throw () [protected,  
inherited]

For use in [\\_Safe\\_sequence](#).

Referenced by \_\_gnu\_debug::Safe\_sequence<\_Sequence >::M\_invalidate\_if(),  
and \_\_gnu\_debug::Safe\_sequence<\_Sequence >::M\_transfer\_from\_if().

**5.244.2.8** void \_\_gnu\_debug::Safe\_sequence\_base::M\_invalidate\_all ( )   
const [inline, inherited]

Invalidates all iterators.

Definition at line 234 of file safe\_base.h.

References \_\_gnu\_debug::Safe\_sequence\_base::\_M\_version.

**5.244.2.9** void \_\_gnu\_debug::Safe\_sequence< deque<\_Tp, \_Allocator >  
>::M\_invalidate\_if ( \_Predicate \_\_pred ) [inherited]

Invalidates all iterators x that reference  
this sequence, are not singular, and for which pred(x) returns true. pred will be  
invoked with the normal iterators nested in the safe ones.

**5.244.2.10** void \_\_gnu\_debug::Safe\_sequence\_base::M\_revalidate\_singular (   
) [protected, inherited]

Revalidates all attached singular iterators. This method may be used to validate  
iterators that were invalidated before (but for some reason, such as an exception, need

## 5.244 std::\_\_debug::deque< \_Tp, \_Allocator > Class Template Reference 1413

to become valid again).

**5.244.2.11** void \_\_gnu\_debug::Safe\_sequence\_base::M\_swap (  
\_Safe\_sequence\_base & \_\_x ) [protected, inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

**5.244.2.12** void \_\_gnu\_debug::Safe\_sequence< deque< \_Tp, \_Allocator  
> >::M\_transfer\_from\_if ( \_Safe\_sequence< deque< \_Tp,  
\_Allocator > > & \_\_from, \_Predicate \_\_pred ) [inherited]

Transfers all iterators *x* that reference *from* sequence, are not singular, and for which *pred(x)* returns *true*. *pred* will be invoked with the normal iterators nested in the safe ones.

### 5.244.3 Member Data Documentation

**5.244.3.1** \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_sequence\_base::M\_  
const\_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 185 of file *safe\_base.h*.

Referenced by \_\_gnu\_debug::Safe\_sequence< \_Sequence >::M\_invalidate\_if(),  
and \_\_gnu\_debug::Safe\_sequence< \_Sequence >::M\_transfer\_from\_if().

**5.244.3.2** \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_sequence\_base::M\_  
iterators [inherited]

The list of mutable iterators that reference this container.

Definition at line 182 of file *safe\_base.h*.

Referenced by \_\_gnu\_debug::Safe\_sequence< \_Sequence >::M\_invalidate\_if(),  
and \_\_gnu\_debug::Safe\_sequence< \_Sequence >::M\_transfer\_from\_if().



## 5.245 `std::__debug::forward_list<_Tp, _Alloc>` Class Template Reference 1414

### 5.244.3.3 `unsigned int __gnu_debug::_Safe_sequence_base::_M_version` [mutable, inherited]

The container version number. This number may never be 0.

Definition at line 188 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`, and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_from_if()`.

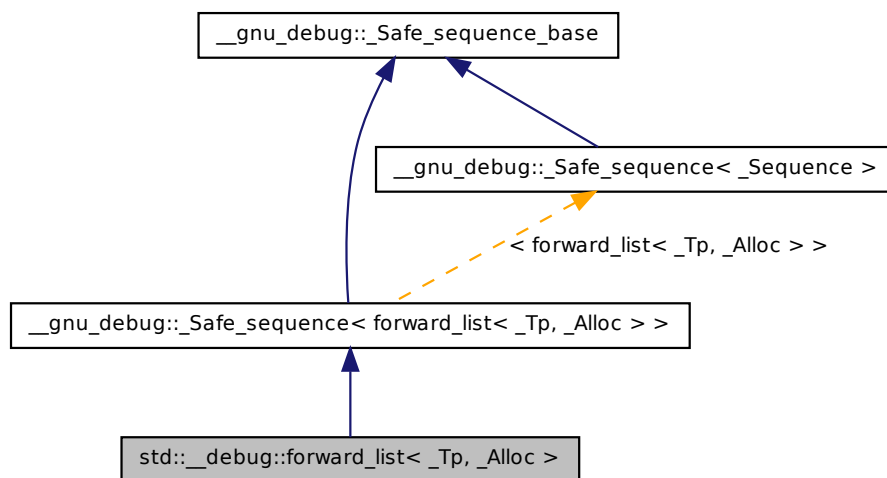
The documentation for this class was generated from the following file:

- [debug/deque](#)

## 5.245 `std::__debug::forward_list<_Tp, _Alloc>` Class Template Reference

Class [std::forward\\_list](#) with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::forward_list<_Tp, _Alloc>`:



### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `__gnu_debug::_Safe_iterator<_Base_const_iterator, forward_list>` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::_Safe_iterator<_Base_iterator, forward_list>` **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `_Base::size_type` **size\_type**
- typedef `_Tp` **value\_type**

### Public Member Functions

- **forward\_list** (const `_Alloc` &\_\_al=\_Alloc())
- **forward\_list** (const `forward_list` &\_\_list, const `_Alloc` &\_\_al)
- **forward\_list** (size\_type \_\_n)
- **forward\_list** (`forward_list` &&\_\_list)
- **forward\_list** (`std::initializer_list`< `_Tp` > \_\_il, const `_Alloc` &\_\_al=\_Alloc())
- **forward\_list** (size\_type \_\_n, const `_Tp` &\_\_value, const `_Alloc` &\_\_al=\_Alloc())
- **forward\_list** (`forward_list` &&\_\_list, const `_Alloc` &\_\_al)
- template<typename `_InputIterator` >  
  **forward\_list** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Alloc` &\_\_al=\_Alloc())
- **forward\_list** (const `forward_list` &\_\_list)
- void **\_M\_attach** (`_Safe_iterator_base` \*\_\_it, bool \_\_constant)
- void **\_M\_attach\_single** (`_Safe_iterator_base` \*\_\_it, bool \_\_constant) throw ()
- `_Base` & **\_M\_base** ()
- const `_Base` & **\_M\_base** () const
- void **\_M\_detach** (`_Safe_iterator_base` \*\_\_it)
- void **\_M\_detach\_single** (`_Safe_iterator_base` \*\_\_it) throw ()
- void **\_M\_invalidate\_all** () const
- void **\_M\_invalidate\_if** (`_Predicate` \_\_pred)
- void **\_M\_transfer\_from\_if** (`_Safe_sequence` &\_\_from, `_Predicate` \_\_pred)
- void **assign** (`std::initializer_list`< `_Tp` > \_\_il)
- template<typename `_InputIterator` >  
  void **assign** (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- void **assign** (size\_type \_\_n, const `_Tp` &\_\_val)
- `iterator` **before\_begin** ()
- `const_iterator` **before\_begin** () const

- `iterator begin ()`
- `const_iterator begin () const`
- `const_iterator cbegin () const`
- `const_iterator cbegin () const`
- `const_iterator cend () const`
- `void clear ()`
- `template<typename... _Args>`  
`iterator emplace_after (const_iterator __pos, _Args &&... __args)`
- `iterator end ()`
- `const_iterator end () const`
- `iterator erase_after (const_iterator __pos)`
- `iterator erase_after (const_iterator __pos, const_iterator __last)`
- `const_reference front () const`
- `reference front ()`
- `iterator insert_after (const_iterator __pos, std::initializer_list<_Tp> __il)`
- `iterator insert_after (const_iterator __pos, size_type __n, const _Tp &__val)`
- `template<typename _InputIterator>`  
`iterator insert_after (const_iterator __pos, _InputIterator __first, _InputIterator __last)`
- `iterator insert_after (const_iterator __pos, const _Tp &__val)`
- `iterator insert_after (const_iterator __pos, _Tp &&__val)`
- `void merge (forward_list &&__list)`
- `template<typename _Comp>`  
`void merge (forward_list &&__list, _Comp __comp)`
- `forward_list & operator= (forward_list &&__list)`
- `forward_list & operator= (std::initializer_list<_Tp> __il)`
- `forward_list & operator= (const forward_list &__list)`
- `void pop_front ()`
- `void remove (const _Tp &__val)`
- `template<typename _Pred>`  
`void remove_if (_Pred __pred)`
- `void resize (size_type __sz)`
- `void resize (size_type __sz, const value_type &__val)`
- `void splice_after (const_iterator __pos, forward_list &&__list, const_iterator __before, const_iterator __last)`
- `void splice_after (const_iterator __pos, forward_list &&__list)`
- `void splice_after (const_iterator __pos, forward_list &&__list, const_iterator __i)`
- `void swap (forward_list &__list)`
- `template<typename _BinPred>`  
`void unique (_BinPred __binary_pred)`
- `void unique ()`

### Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

### Protected Member Functions

- `void _M_detach_all ()`
- `void _M_detach_singular ()`
- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`
- `void _M_revalidate_singular ()`
- `void _M_swap (_Safe_sequence_base &__x)`

#### 5.245.1 Detailed Description

`template<typename _Tp, typename _Alloc = std::allocator<_Tp>> class std::_debug::forward_list<_Tp, _Alloc>`

Class `std::forward_list` with safety/checking/debug instrumentation.

Definition at line 44 of file `debug/forward_list`.

#### 5.245.2 Member Function Documentation

**5.245.2.1** `void __gnu_debug::_Safe_sequence_base::_M_attach (   
_Safe_iterator_base * __it, bool __constant ) [inherited]`

Attach an iterator to this sequence.

**5.245.2.2** `void __gnu_debug::_Safe_sequence_base::_M_attach_single   
( _Safe_iterator_base * __it, bool __constant ) throw ()   
[inherited]`

Likewise but not thread safe.

**5.245.2.3** `void __gnu_debug::_Safe_sequence_base::_M_detach (   
_Safe_iterator_base * __it ) [inherited]`

Detach an iterator from this sequence

## 5.245 std::\_\_debug::forward\_list<\_Tp, \_Alloc> Class Template Reference 1418

**5.245.2.4** void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach\_all ( )  
[protected, inherited]

Detach all iterators, leaving them singular.

Referenced by \_\_gnu\_debug::Safe\_sequence\_base::~~Safe\_sequence\_base().

**5.245.2.5** void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach\_single (   
\_Safe\_iterator\_base \* \_\_it ) throw () [inherited]

Likewise but not thread safe.

**5.245.2.6** void \_\_gnu\_debug::Safe\_sequence\_base::M\_detach\_singular ( )  
[protected, inherited]

Detach all singular iterators.

### Postcondition

for all iterators *i* attached to this sequence, *i*->\_M\_version == \_M\_version.

**5.245.2.7** \_\_gnu\_cxx::mutex& \_\_gnu\_debug::Safe\_sequence\_  
base::M\_get\_mutex ( ) throw () [protected,  
inherited]

For use in [\\_Safe\\_sequence](#).

Referenced by \_\_gnu\_debug::Safe\_sequence<\_Sequence>::M\_invalidate\_if(),  
and \_\_gnu\_debug::Safe\_sequence<\_Sequence>::M\_transfer\_from\_if().

**5.245.2.8** void \_\_gnu\_debug::Safe\_sequence\_base::M\_invalidate\_all ( )  
const [inline, inherited]

Invalidates all iterators.

Definition at line 234 of file safe\_base.h.

References \_\_gnu\_debug::Safe\_sequence\_base::M\_version.

**5.245.2.9** void \_\_gnu\_debug::Safe\_sequence<forward\_list<\_Tp, \_Alloc>  
>::M\_invalidate\_if ( \_Predicate \_\_pred ) [inherited]

Invalidates all iterators *x* that reference  
this sequence, are not singular, and for which *pred*(*x*) returns true. *pred* will be  
invoked with the normal iterators nested in the safe ones.

## 5.245 `std::__debug::forward_list<_Tp, _Alloc>` Class Template Reference 1419

**5.245.2.10** `void __gnu_debug::Safe_sequence_base::_M_revalidate_singular ( ) [protected, inherited]`

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

**5.245.2.11** `void __gnu_debug::Safe_sequence_base::_M_swap ( _Safe_sequence_base & __x ) [protected, inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

**5.245.2.12** `void __gnu_debug::Safe_sequence< forward_list<_Tp, _Alloc> >::_M_transfer_from_if ( _Safe_sequence< forward_list<_Tp, _Alloc> > & __from, _Predicate __pred ) [inherited]`

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `pred(x)` returns `true`. `pred` will be invoked with the normal iterators nested in the safe ones.

### 5.245.3 Member Data Documentation

**5.245.3.1** `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::_M_const_iterators [inherited]`

The list of constant iterators that reference this container.

Definition at line 185 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::_M_invalidate_if()`, and `__gnu_debug::Safe_sequence< _Sequence >::_M_transfer_from_if()`.

**5.245.3.2** `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::_M_iterators [inherited]`

The list of mutable iterators that reference this container.

Definition at line 182 of file `safe_base.h`.

## 5.246 `std::__debug::list<_Tp, _Allocator>` Class Template Reference 1420

---

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if()`, and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_from_if()`.

### 5.245.3.3 `unsigned int __gnu_debug::_Safe_sequence_base::_M_version` [mutable, inherited]

The container version number. This number may never be 0.

Definition at line 188 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`, and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_from_if()`.

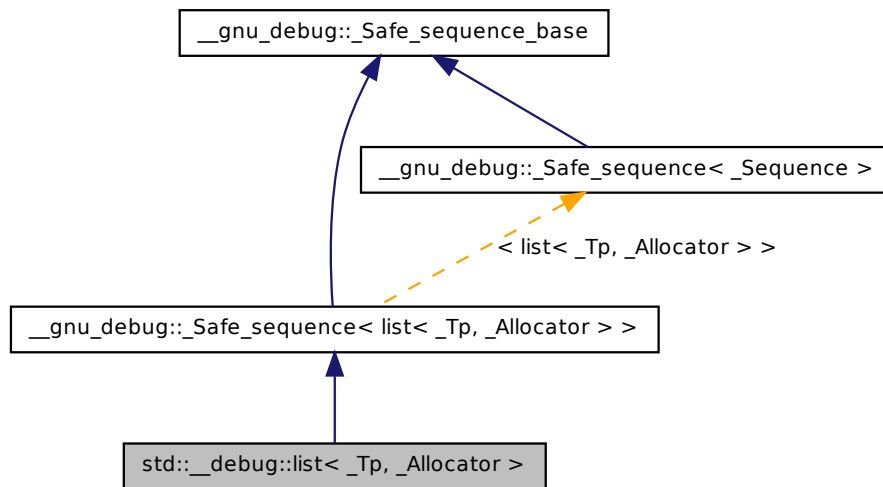
The documentation for this class was generated from the following file:

- [debug/forward\\_list](#)

## 5.246 `std::__debug::list<_Tp, _Allocator>` Class Template Reference

Class [std::list](#) with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::list< _Tp, _Allocator >`:



### Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::Safe_iterator< _Base_const_iterator, list >` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::Safe_iterator< _Base_iterator, list >` **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Tp` **value\_type**

### Public Member Functions

- **list** (`const _Allocator &__a=_Allocator()`)



- **list** (size\_type \_\_n)
- template<class \_InputIterator >  
  **list** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Allocator &\_\_a=\_Allocator())
- **list** (initializer\_list< value\_type > \_\_l, const allocator\_type &\_\_a=allocator\_type())
- **list** (const [list](#) &\_\_x)
- **list** (size\_type \_\_n, const \_Tp &\_\_value, const \_Allocator &\_\_a=\_Allocator())
- **list** (const [\\_Base](#) &\_\_x)
- **list** ([list](#) &&\_\_x)
- void [\\_M\\_attach](#) (\_Safe\_iterator\_base \*\_\_it, bool \_\_constant)
- void [\\_M\\_attach\\_single](#) (\_Safe\_iterator\_base \*\_\_it, bool \_\_constant) throw ()
- [\\_Base](#) & [\\_M\\_base](#) ()
- const [\\_Base](#) & [\\_M\\_base](#) () const
- void [\\_M\\_detach](#) (\_Safe\_iterator\_base \*\_\_it)
- void [\\_M\\_detach\\_single](#) (\_Safe\_iterator\_base \*\_\_it) throw ()
- void [\\_M\\_invalidate\\_all](#) () const
- void [\\_M\\_invalidate\\_if](#) (\_Predicate \_\_pred)
- void [\\_M\\_transfer\\_from\\_if](#) (\_Safe\_sequence &\_\_from, \_Predicate \_\_pred)
- void **assign** (initializer\_list< value\_type > \_\_l)
- template<class \_InputIterator >  
  void **assign** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void **assign** (size\_type \_\_n, const \_Tp &\_\_t)
- const\_reference **back** () const
- reference **back** ()
- [iterator](#) **begin** ()
- const\_iterator **begin** () const
- const\_iterator **cbegin** () const
- const\_iterator **cend** () const
- void **clear** ()
- const\_reverse\_iterator **crbegin** () const
- const\_reverse\_iterator **crend** () const
- template<typename... \_Args>  
  [iterator](#) **emplace** ([iterator](#) \_\_position, \_Args &&...\_\_args)
- [iterator](#) **end** ()
- const\_iterator **end** () const
- [iterator](#) **erase** ([iterator](#) \_\_position)
- [iterator](#) **erase** ([iterator](#) \_\_position, [iterator](#) \_\_last)
- const\_reference **front** () const
- reference **front** ()
- [iterator](#) **insert** ([iterator](#) \_\_position, const \_Tp &\_\_x)
- [iterator](#) **insert** ([iterator](#) \_\_position, \_Tp &&\_\_x)
- void **insert** ([iterator](#) \_\_p, initializer\_list< value\_type > \_\_l)

- `template<class _InputIterator>`  
`void insert (iterator __position, _InputIterator __first, _InputIterator __last)`
- `void insert (iterator __position, size_type __n, const _Tp &__x)`
- `template<class _Compare>`  
`void merge (list &&__x, _Compare __comp)`
- `template<typename _Compare>`  
`void merge (list &__x, _Compare __comp)`
- `void merge (list &&__x)`
- `void merge (list &__x)`
- `list & operator= (const list &__x)`
- `list & operator= (list &&__x)`
- `list & operator= (initializer_list< value_type > __l)`
- `void pop_back ()`
- `void pop_front ()`
- `reverse_iterator rbegin ()`
- `const_reverse_iterator rbegin () const`
- `void remove (const _Tp &__value)`
- `template<class _Predicate>`  
`void remove_if (_Predicate __pred)`
- `reverse_iterator rend ()`
- `const_reverse_iterator rend () const`
- `void resize (size_type __sz, const _Tp &__c)`
- `void resize (size_type __sz)`
- `template<typename _StrictWeakOrdering>`  
`void sort (_StrictWeakOrdering __pred)`
- `void sort ()`
- `void splice (iterator __position, list &&__x)`
- `void splice (iterator __position, list &&__x, iterator __first, iterator __last)`
- `void splice (iterator __position, list &&__x, iterator __i)`
- `void splice (iterator __position, list &__x, iterator __i)`
- `void splice (iterator __position, list &__x, iterator __first, iterator __last)`
- `void splice (iterator __position, list &__x)`
- `void swap (list &__x)`
- `void unique ()`
- `template<class _BinaryPredicate>`  
`void unique (_BinaryPredicate __binary_pred)`

### Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

**Protected Member Functions**

- void [\\_M\\_detach\\_all](#) ()
- void [\\_M\\_detach\\_singular](#) ()
- [\\_\\_gnu\\_cxx::\\_\\_mutex](#) & [\\_M\\_get\\_mutex](#) () throw ()
- void [\\_M\\_revalidate\\_singular](#) ()
- void [\\_M\\_swap](#) (\_Safe\_sequence\_base &\_\_x)

**5.246.1 Detailed Description**

template<typename \_Tp, typename \_Allocator = std::allocator<\_Tp>> class  
std::\_\_debug::list<\_Tp, \_Allocator>

Class [std::list](#) with safety/checking/debug instrumentation.

Definition at line 43 of file debug/list.

**5.246.2 Member Function Documentation**

**5.246.2.1** void [\\_\\_gnu\\_debug::Safe\\_sequence\\_base::M\\_attach](#) (  
\_Safe\_iterator\_base \* *\_\_it*, bool *\_\_constant* ) [*inherited*]

Attach an iterator to this sequence.

**5.246.2.2** void [\\_\\_gnu\\_debug::Safe\\_sequence\\_base::M\\_attach\\_single](#)  
( \_Safe\_iterator\_base \* *\_\_it*, bool *\_\_constant* ) throw ()  
[*inherited*]

Likewise but not thread safe.

**5.246.2.3** void [\\_\\_gnu\\_debug::Safe\\_sequence\\_base::M\\_detach](#) (  
\_Safe\_iterator\_base \* *\_\_it* ) [*inherited*]

Detach an iterator from this sequence

**5.246.2.4** void [\\_\\_gnu\\_debug::Safe\\_sequence\\_base::M\\_detach\\_all](#) ( )  
[*protected*, *inherited*]

Detach all iterators, leaving them singular.

Referenced by [\\_\\_gnu\\_debug::Safe\\_sequence\\_base::~~Safe\\_sequence\\_base](#) ().

**5.246.2.5** `void __gnu_debug::_Safe_sequence_base::_M_detach_single (`  
`_Safe_iterator_base * __it ) throw () [inherited]`

Likewise but not thread safe.

**5.246.2.6** `void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( )`  
`[protected, inherited]`

Detach all singular iterators.

**Postcondition**

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

**5.246.2.7** `__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_-`  
`base::_M_get_mutex ( ) throw () [protected,`  
`inherited]`

For use in [\\_Safe\\_sequence](#).

Referenced by `__gnu_debug::_Safe_sequence<_Sequence >::_M_invalidate_if()`,  
and `__gnu_debug::_Safe_sequence<_Sequence >::_M_transfer_from_if()`.

**5.246.2.8** `void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( )`  
`const [inline, inherited]`

Invalidates all iterators.

Definition at line 234 of file `safe_base.h`.

References `__gnu_debug::_Safe_sequence_base::_M_version`.

**5.246.2.9** `void __gnu_debug::_Safe_sequence<list<_Tp, _Allocator >`  
`>::_M_invalidate_if ( _Predicate __pred ) [inherited]`

Invalidates all iterators `x` that reference  
this sequence, are not singular, and for which `pred(x)` returns `true`. `pred` will be  
invoked with the normal iterators nested in the safe ones.

**5.246.2.10** `void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular (`  
`) [protected, inherited]`

Revalidates all attached singular iterators. This method may be used to validate  
iterators that were invalidated before (but for some reason, such as an exception, need

to become valid again).

**5.246.2.11** void \_\_gnu\_debug::Safe\_sequence\_base::M\_swap (  
\_Safe\_sequence\_base & \_\_x ) [protected, inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

**5.246.2.12** void \_\_gnu\_debug::Safe\_sequence< list< \_Tp, \_Allocator >  
>::M\_transfer\_from\_if ( \_Safe\_sequence< list< \_Tp, \_Allocator  
> & \_\_from, \_Predicate \_\_pred ) [inherited]

Transfers all iterators *x* that reference *from* sequence, are not singular, and for which *pred*(*x*) returns *true*. *pred* will be invoked with the normal iterators nested in the safe ones.

### 5.246.3 Member Data Documentation

**5.246.3.1** \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_sequence\_base::M\_  
const\_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 185 of file *safe\_base.h*.

Referenced by \_\_gnu\_debug::Safe\_sequence< \_Sequence >::M\_invalidate\_if(),  
and \_\_gnu\_debug::Safe\_sequence< \_Sequence >::M\_transfer\_from\_if().

**5.246.3.2** \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_sequence\_base::M\_  
iterators [inherited]

The list of mutable iterators that reference this container.

Definition at line 182 of file *safe\_base.h*.

Referenced by \_\_gnu\_debug::Safe\_sequence< \_Sequence >::M\_invalidate\_if(),  
and \_\_gnu\_debug::Safe\_sequence< \_Sequence >::M\_transfer\_from\_if().

## 5.247 `std::__debug::map< _Key, _Tp, _Compare, _Allocator >` Class Template Reference 1427

### 5.246.3.3 `unsigned int __gnu_debug::_Safe_sequence_base::_M_version` [mutable, inherited]

The container version number. This number may never be 0.

Definition at line 188 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if()`.

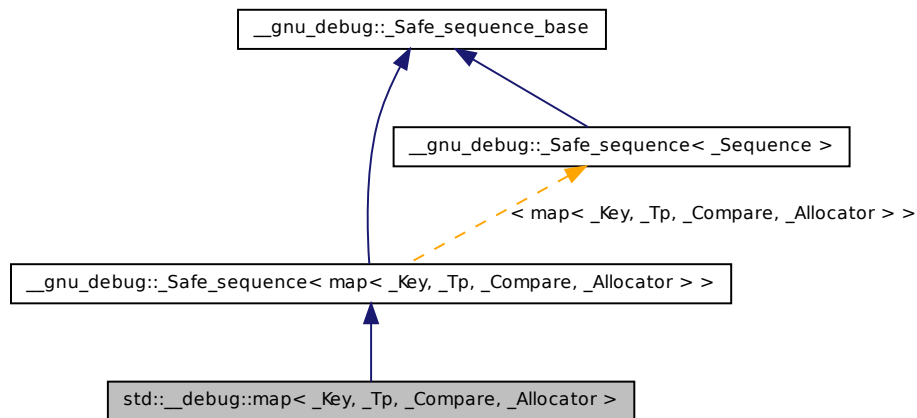
The documentation for this class was generated from the following file:

- [debug/list](#)

## 5.247 `std::__debug::map< _Key, _Tp, _Compare, _Allocator >` Class Template Reference

Class [std::map](#) with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::map< _Key, _Tp, _Compare, _Allocator >`:



### Public Types

- `typedef _Allocator allocator_type`

- typedef `__gnu_debug::_Safe_iterator`< typename `_Base::const_iterator`, `map` > `const_iterator`
- typedef `_Base::const_pointer` `const_pointer`
- typedef `_Base::const_reference` `const_reference`
- typedef `std::reverse_iterator`< `const_iterator` > `const_reverse_iterator`
- typedef `_Base::difference_type` `difference_type`
- typedef `__gnu_debug::_Safe_iterator`< typename `_Base::iterator`, `map` > `iterator`
- typedef `_Compare` `key_compare`
- typedef `_Key` `key_type`
- typedef `_Tp` `mapped_type`
- typedef `_Base::pointer` `pointer`
- typedef `_Base::reference` `reference`
- typedef `std::reverse_iterator`< `iterator` > `reverse_iterator`
- typedef `_Base::size_type` `size_type`
- typedef `std::pair`< `const _Key`, `_Tp` > `value_type`

#### Public Member Functions

- **map** (const `_Compare` &\_\_comp=\_Compare(), const `_Allocator` &\_\_a=\_Allocator())
- template<typename `_InputIterator` >  
**map** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Compare` &\_\_comp=\_Compare(), const `_Allocator` &\_\_a=\_Allocator())
- **map** (const `_Base` &\_\_x)
- **map** (`map` &&\_\_x)
- **map** (const `map` &\_\_x)
- **map** (`initializer_list`< `value_type` > \_\_l, const `_Compare` &\_\_c=\_Compare(), const `allocator_type` &\_\_a=allocator\_type())
- void `_M_attach` (`_Safe_iterator_base` \*\_\_it, bool \_\_constant)
- void `_M_attach_single` (`_Safe_iterator_base` \*\_\_it, bool \_\_constant) throw ()
- `_Base` & `_M_base` ()
- const `_Base` & `_M_base` () const
- void `_M_detach` (`_Safe_iterator_base` \*\_\_it)
- void `_M_detach_single` (`_Safe_iterator_base` \*\_\_it) throw ()
- void `_M_invalidate_all` () const
- void `_M_invalidate_if` (`_Predicate` \_\_pred)
- void `_M_transfer_from_if` (`_Safe_sequence` &\_\_from, `_Predicate` \_\_pred)
- `const_iterator` **begin** () const
- `iterator` **begin** ()
- `const_iterator` **cbegin** () const
- `const_iterator` **cend** () const
- void **clear** ()

- `const_reverse_iterator` **crbegin** () const
- `const_reverse_iterator` **crend** () const
- `iterator` **end** ()
- `const_iterator` **end** () const
- `std::pair< iterator, iterator >` **equal\_range** (const key\_type &\_\_x)
- `std::pair< const_iterator, const_iterator >` **equal\_range** (const key\_type &\_\_x) const
- `iterator` **erase** (const\_iterator \_\_first, const\_iterator \_\_last)
- `iterator` **erase** (const\_iterator \_\_position)
- size\_type **erase** (const key\_type &\_\_x)
- `iterator` **find** (const key\_type &\_\_x)
- `const_iterator` **find** (const key\_type &\_\_x) const
- void **insert** (std::initializer\_list< value\_type > \_\_list)
- `std::pair< iterator, bool >` **insert** (const value\_type &\_\_x)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_convertible<\_Pair, value\_type>::value>::type>  
`std::pair< iterator, bool >` **insert** (\_Pair &&\_\_x)
- `iterator` **insert** (const\_iterator \_\_position, const value\_type &\_\_x)
- template<typename \_InputIterator >  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_convertible<\_Pair, value\_type>::value>::type>  
`iterator` **insert** (const\_iterator \_\_position, \_Pair &&\_\_x)
- `iterator` **lower\_bound** (const key\_type &\_\_x)
- `const_iterator` **lower\_bound** (const key\_type &\_\_x) const
- `map` & **operator=** (const `map` &\_\_x)
- `map` & **operator=** (`map` &&\_\_x)
- `map` & **operator=** (initializer\_list< value\_type > \_\_l)
- `reverse_iterator` **rbegin** ()
- `const_reverse_iterator` **rbegin** () const
- `reverse_iterator` **rend** ()
- `const_reverse_iterator` **rend** () const
- void **swap** (`map` &\_\_x)
- `iterator` **upper\_bound** (const key\_type &\_\_x)
- `const_iterator` **upper\_bound** (const key\_type &\_\_x) const

#### Public Attributes

- \_Safe\_iterator\_base \* `_M_const_iterators`
- \_Safe\_iterator\_base \* `_M_iterators`
- unsigned int `_M_version`



### Protected Member Functions

- `void _M_detach_all()`
- `void _M_detach_singular()`
- `__gnu_cxx::__mutex & _M_get_mutex() throw()`
- `void _M_revalidate_singular()`
- `void _M_swap(_Safe_sequence_base &__x)`

#### 5.247.1 Detailed Description

`template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<std::pair<const _Key, _Tp>>>> class std::__debug::map<_Key, _Tp, _Compare, _Allocator>`

Class `std::map` with safety/checking/debug instrumentation.

Definition at line 44 of file `debug/map.h`.

#### 5.247.2 Member Function Documentation

5.247.2.1 `void __gnu_debug::_Safe_sequence_base::_M_attach ( _Safe_iterator_base * __it, bool __constant ) [inherited]`

Attach an iterator to this sequence.

5.247.2.2 `void __gnu_debug::_Safe_sequence_base::_M_attach_single ( _Safe_iterator_base * __it, bool __constant ) throw () [inherited]`

Likewise but not thread safe.

5.247.2.3 `void __gnu_debug::_Safe_sequence_base::_M_detach ( _Safe_iterator_base * __it ) [inherited]`

Detach an iterator from this sequence

5.247.2.4 `void __gnu_debug::_Safe_sequence_base::_M_detach_all ( ) [protected, inherited]`

Detach all iterators, leaving them singular.

Referenced by `__gnu_debug::_Safe_sequence_base::~_Safe_sequence_base()`.

5.247.2.5 `void __gnu_debug::_Safe_sequence_base::_M_detach_single (   
_Safe_iterator_base * __it ) throw () [inherited]`

Likewise but not thread safe.

5.247.2.6 `void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( )   
[protected, inherited]`

Detach all singular iterators.

#### Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

5.247.2.7 `__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_  
base::_M_get_mutex ( ) throw () [protected,  
inherited]`

For use in [\\_Safe\\_sequence](#).

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`,  
and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if()`.

5.247.2.8 `void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( )   
const [inline, inherited]`

Invalidates all iterators.

Definition at line 234 of file `safe_base.h`.

References `__gnu_debug::_Safe_sequence_base::_M_version`.

5.247.2.9 `void __gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare,  
_Allocator > >::_M_invalidate_if ( _Predicate __pred )   
[inherited]`

Invalidates all iterators `x` that reference  
this sequence, are not singular, and for which `pred(x)` returns `true`. `pred` will be  
invoked with the normal iterators nested in the safe ones.

**5.247.2.10** void \_\_gnu\_debug::Safe\_sequence\_base::\_M\_revalidate\_singular ( ) [protected, inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

**5.247.2.11** void \_\_gnu\_debug::Safe\_sequence\_base::\_M\_swap ( \_Safe\_sequence\_base & \_\_x ) [protected, inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

**5.247.2.12** void \_\_gnu\_debug::Safe\_sequence< map<\_Key, \_Tp, \_Compare, \_Allocator> >::\_M\_transfer\_from\_if ( \_Safe\_sequence< map<\_Key, \_Tp, \_Compare, \_Allocator> > & \_\_from, \_Predicate \_\_pred ) [inherited]

Transfers all iterators *x* that reference *from* sequence, are not singular, and for which *pred(x)* returns true. *pred* will be invoked with the normal iterators nested in the safe ones.

### 5.247.3 Member Data Documentation

**5.247.3.1** \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_sequence\_base::\_M\_const\_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 185 of file safe\_base.h.

Referenced by \_\_gnu\_debug::Safe\_sequence<\_Sequence>::\_M\_invalidate\_if(), and \_\_gnu\_debug::Safe\_sequence<\_Sequence>::\_M\_transfer\_from\_if().

**5.247.3.2** \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_sequence\_base::\_M\_iterators [inherited]

The list of mutable iterators that reference this container.

## 5.248 `std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >` Class Template Reference 1433

Definition at line 182 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if()`.

### 5.247.3.3 `unsigned int __gnu_debug::_Safe_sequence_base::_M_version` [mutable, inherited]

The container version number. This number may never be 0.

Definition at line 188 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if()`.

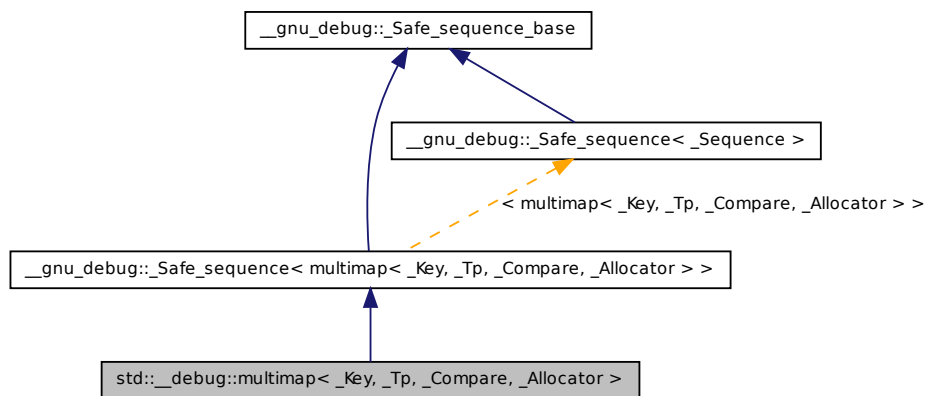
The documentation for this class was generated from the following file:

- [debug/map.h](#)

## 5.248 `std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >` Class Template Reference

Class `std::multimap` with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >`:



## Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::_Safe_iterator<_Base_const_iterator, multimap >` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator<const_iterator>` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::_Safe_iterator<_Base_iterator, multimap >` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Tp` **mapped\_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator<iterator>` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `std::pair<const _Key, _Tp>` **value\_type**

## Public Member Functions

- **multimap** (const `_Compare` &\_\_comp=\_Compare(), const `_Allocator` &\_\_a=\_Allocator())
- template<typename `_InputIterator` >  
**multimap** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Compare` &\_\_comp=\_Compare(), const `_Allocator` &\_\_a=\_Allocator())
- **multimap** (const `_Base` &\_\_x)
- **multimap** (`multimap` &&\_\_x)
- **multimap** (const `multimap` &\_\_x)
- **multimap** (`initializer_list`< `value_type` > \_\_l, const `_Compare` &\_\_c=\_Compare(), const `allocator_type` &\_\_a=allocator\_type())
- void `_M_attach` (`_Safe_iterator_base` \*\_\_it, bool \_\_constant)
- void `_M_attach_single` (`_Safe_iterator_base` \*\_\_it, bool \_\_constant) throw ()
- `_Base` & `_M_base` ()
- const `_Base` & `_M_base` () const
- void `_M_detach` (`_Safe_iterator_base` \*\_\_it)
- void `_M_detach_single` (`_Safe_iterator_base` \*\_\_it) throw ()
- void `_M_invalidate_all` () const
- void `_M_invalidate_if` (`_Predicate` \_\_pred)
- void `_M_transfer_from_if` (`_Safe_sequence` &\_\_from, `_Predicate` \_\_pred)
- `const_iterator` **begin** () const
- `iterator` **begin** ()
- `const_iterator` **cbegin** () const

- `const_iterator cend` () const
- void `clear` ()
- `const_reverse_iterator crbegin` () const
- `const_reverse_iterator crend` () const
- `iterator end` ()
- `const_iterator end` () const
- `std::pair< iterator, iterator > equal_range` (const key\_type &\_\_x)
- `std::pair< const_iterator, const_iterator > equal_range` (const key\_type &\_\_x) const
- `iterator erase` (const\_iterator \_\_first, const\_iterator \_\_last)
- `iterator erase` (const\_iterator \_\_position)
- size\_type `erase` (const key\_type &\_\_x)
- `iterator find` (const key\_type &\_\_x)
- `const_iterator find` (const key\_type &\_\_x) const
- void `insert` (std::initializer\_list< value\_type > \_\_list)
- `iterator insert` (const value\_type &\_\_x)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_convertible<\_Pair, value\_type>::value>::type>  
`iterator insert` (\_Pair &&\_\_x)
- `iterator insert` (const\_iterator \_\_position, const value\_type &\_\_x)
- template<typename \_InputIterator >  
void `insert` (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_convertible<\_Pair, value\_type>::value>::type>  
`iterator insert` (const\_iterator \_\_position, \_Pair &&\_\_x)
- `iterator lower_bound` (const key\_type &\_\_x)
- `const_iterator lower_bound` (const key\_type &\_\_x) const
- `multimap & operator=` (const multimap &\_\_x)
- `multimap & operator=` (multimap &&\_\_x)
- `multimap & operator=` (initializer\_list< value\_type > \_\_l)
- `reverse_iterator rbegin` ()
- `const_reverse_iterator rbegin` () const
- `reverse_iterator rend` ()
- `const_reverse_iterator rend` () const
- void `swap` (multimap &\_\_x)
- `iterator upper_bound` (const key\_type &\_\_x)
- `const_iterator upper_bound` (const key\_type &\_\_x) const

#### Public Attributes

- \_Safe\_iterator\_base \* `_M_const_iterators`
- \_Safe\_iterator\_base \* `_M_iterators`
- unsigned int `_M_version`

### Protected Member Functions

- `void _M_detach_all()`
- `void _M_detach_singular()`
- `__gnu_cxx::__mutex & _M_get_mutex() throw()`
- `void _M_revalidate_singular()`
- `void _M_swap(_Safe_sequence_base &__x)`

#### 5.248.1 Detailed Description

`template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<std::pair<const _Key, _Tp>>>>`  
`class std::__debug::multimap<_Key,_Tp,_Compare,_Allocator>`

Class `std::multimap` with safety/checking/debug instrumentation.

Definition at line 44 of file `debug/multimap.h`.

#### 5.248.2 Member Function Documentation

5.248.2.1 `void __gnu_debug::Safe_sequence_base::_M_attach (`  
`_Safe_iterator_base * __it, bool __constant ) [inherited]`

Attach an iterator to this sequence.

5.248.2.2 `void __gnu_debug::Safe_sequence_base::_M_attach_single`  
`( _Safe_iterator_base * __it, bool __constant ) throw ()`  
`[inherited]`

Likewise but not thread safe.

5.248.2.3 `void __gnu_debug::Safe_sequence_base::_M_detach (`  
`_Safe_iterator_base * __it ) [inherited]`

Detach an iterator from this sequence

5.248.2.4 `void __gnu_debug::Safe_sequence_base::_M_detach_all ( )`  
`[protected, inherited]`

Detach all iterators, leaving them singular.

Referenced by `__gnu_debug::Safe_sequence_base::~~Safe_sequence_base()`.

5.248.2.5 `void __gnu_debug::_Safe_sequence_base::_M_detach_single (`  
`_Safe_iterator_base * __it ) throw () [inherited]`

Likewise but not thread safe.

5.248.2.6 `void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( )`  
`[protected, inherited]`

Detach all singular iterators.

**Postcondition**

for all iterators *i* attached to this sequence, `i->_M_version == _M_version`.

5.248.2.7 `__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_-`  
`base::_M_get_mutex ( ) throw () [protected,`  
`inherited]`

For use in [\\_Safe\\_sequence](#).

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if()`,  
and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_from_if()`.

5.248.2.8 `void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( )`  
`const [inline, inherited]`

Invalidates all iterators.

Definition at line 234 of file `safe_base.h`.

References `__gnu_debug::_Safe_sequence_base::_M_version`.

5.248.2.9 `void __gnu_debug::_Safe_sequence<multimap<_Key, _Tp,`  
`_Compare, _Allocator>>::_M_invalidate_if ( _Predicate __pred )`  
`[inherited]`

Invalidates all iterators *x* that reference  
this sequence, are not singular, and for which `pred(x)` returns `true`. `pred` will be  
invoked with the normal iterators nested in the safe ones.



**5.248.2.10** `void __gnu_debug::Safe_sequence_base::_M_revalidate_singular ( )` `[protected, inherited]`

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

**5.248.2.11** `void __gnu_debug::Safe_sequence_base::_M_swap ( _Safe_sequence_base & __x )` `[protected, inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

**5.248.2.12** `void __gnu_debug::Safe_sequence< multimap<_Key,_Tp,_Compare,_Allocator> >::_M_transfer_from_if ( _Safe_sequence< multimap<_Key,_Tp,_Compare,_Allocator> > & __from, _Predicate __pred )` `[inherited]`

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `pred(x)` returns `true`. `pred` will be invoked with the normal iterators nested in the safe ones.

### 5.248.3 Member Data Documentation

**5.248.3.1** `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::_M_const_iterators` `[inherited]`

The list of constant iterators that reference this container.

Definition at line 185 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence<_Sequence>::_M_invalidate_if()`, and `__gnu_debug::Safe_sequence<_Sequence>::_M_transfer_from_if()`.

**5.248.3.2** `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::_M_iterators` `[inherited]`

The list of mutable iterators that reference this container.

## 5.249 `std::__debug::multiset< _Key, _Compare, _Allocator >` Class Template Reference 1439

Definition at line 182 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if()`.

### 5.248.3.3 `unsigned int __gnu_debug::_Safe_sequence_base::_M_version` [mutable, inherited]

The container version number. This number may never be 0.

Definition at line 188 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if()`.

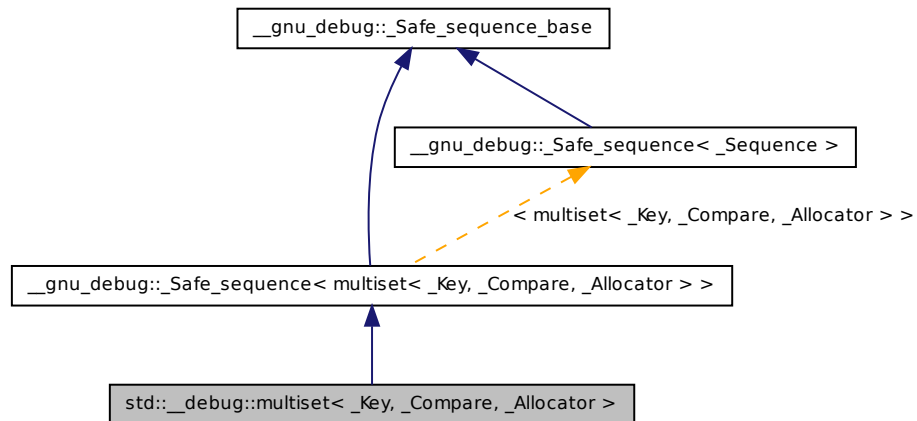
The documentation for this class was generated from the following file:

- [debug/multimap.h](#)

## 5.249 `std::__debug::multiset< _Key, _Compare, _Allocator >` Class Template Reference

Class [std::multiset](#) with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::multiset< _Key, _Compare, _Allocator >`:



## Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::_Safe_iterator<_Base_const_iterator, multiset >` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator<const_iterator>` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::_Safe_iterator<_Base_iterator, multiset>` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator<iterator>` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Compare` **value\_compare**
- typedef `_Key` **value\_type**

## Public Member Functions

- **multiset** (`const _Compare &__comp=_Compare()`, `const _Allocator &__a=_Allocator()`)
- `template<typename _InputIterator>`  
**multiset** (`_InputIterator __first, _InputIterator __last, const _Compare &__comp=_Compare()`, `const _Allocator &__a=_Allocator()`)
- **multiset** (`const _Base &__x`)
- **multiset** (`multiset &&__x`)
- **multiset** (`const multiset &__x`)
- **multiset** (`initializer_list<value_type> > __l, const _Compare &__comp=_Compare()`, `const allocator_type &__a=allocator_type()`)
- `void _M_attach (_Safe_iterator_base *__it, bool __constant)`
- `void _M_attach_single (_Safe_iterator_base *__it, bool __constant) throw ()`
- `_Base & _M_base ()`
- `const _Base & _M_base () const`
- `void _M_detach (_Safe_iterator_base *__it)`
- `void _M_detach_single (_Safe_iterator_base *__it) throw ()`
- `void _M_invalidate_all () const`
- `void _M_invalidate_if (_Predicate __pred)`
- `void _M_transfer_from_if (_Safe_sequence &__from, _Predicate __pred)`
- `const_iterator begin () const`
- `iterator begin ()`
- `const_iterator cbegin () const`

- `const_iterator cend` () const
- void `clear` ()
- `const_reverse_iterator crbegin` () const
- `const_reverse_iterator crend` () const
- `iterator end` ()
- `const_iterator end` () const
- `std::pair< iterator, iterator > equal_range` (const key\_type &\_\_x)
- `std::pair< const_iterator, const_iterator > equal_range` (const key\_type &\_\_x)  
const
- `iterator erase` (const\_iterator \_\_first, const\_iterator \_\_last)
- `iterator erase` (const\_iterator \_\_position)
- size\_type `erase` (const key\_type &\_\_x)
- `iterator find` (const key\_type &\_\_x)
- `const_iterator find` (const key\_type &\_\_x) const
- `iterator insert` (const\_iterator \_\_position, const value\_type &\_\_x)
- `iterator insert` (const value\_type &\_\_x)
- `iterator insert` (value\_type &&\_\_x)
- `iterator insert` (const\_iterator \_\_position, value\_type &&\_\_x)
- void `insert` (initializer\_list< value\_type > \_\_l)
- template<typename \_InputIterator >  
void `insert` (\_InputIterator \_\_first, \_InputIterator \_\_last)
- `iterator lower_bound` (const key\_type &\_\_x)
- `const_iterator lower_bound` (const key\_type &\_\_x) const
- `multiset & operator=` (const multiset &\_\_x)
- `multiset & operator=` (multiset &&\_\_x)
- `multiset & operator=` (initializer\_list< value\_type > \_\_l)
- `reverse_iterator rbegin` ()
- `const_reverse_iterator rbegin` () const
- `reverse_iterator rend` ()
- `const_reverse_iterator rend` () const
- void `swap` (multiset &\_\_x)
- `iterator upper_bound` (const key\_type &\_\_x)
- `const_iterator upper_bound` (const key\_type &\_\_x) const

#### Public Attributes

- \_Safe\_iterator\_base \* `_M_const_iterators`
- \_Safe\_iterator\_base \* `_M_iterators`
- unsigned int `_M_version`

### Protected Member Functions

- `void _M_detach_all()`
- `void _M_detach_singular()`
- `__gnu_cxx::__mutex & _M_get_mutex() throw()`
- `void _M_revalidate_singular()`
- `void _M_swap(_Safe_sequence_base &__x)`

#### 5.249.1 Detailed Description

`template<typename _Key, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<_Key>> class std::__debug::multiset<_Key, _Compare, _Allocator>`

Class `std::multiset` with safety/checking/debug instrumentation.

Definition at line 44 of file `debug/multiset.h`.

#### 5.249.2 Member Function Documentation

**5.249.2.1** `void __gnu_debug::Safe_sequence_base::_M_attach ( _Safe_iterator_base * __it, bool __constant ) [inherited]`

Attach an iterator to this sequence.

**5.249.2.2** `void __gnu_debug::Safe_sequence_base::_M_attach_single ( _Safe_iterator_base * __it, bool __constant ) throw () [inherited]`

Likewise but not thread safe.

**5.249.2.3** `void __gnu_debug::Safe_sequence_base::_M_detach ( _Safe_iterator_base * __it ) [inherited]`

Detach an iterator from this sequence

**5.249.2.4** `void __gnu_debug::Safe_sequence_base::_M_detach_all ( ) [protected, inherited]`

Detach all iterators, leaving them singular.

Referenced by `__gnu_debug::Safe_sequence_base::~~Safe_sequence_base()`.

5.249.2.5 `void __gnu_debug::_Safe_sequence_base::_M_detach_single (   
_Safe_iterator_base * __it ) throw () [inherited]`

Likewise but not thread safe.

5.249.2.6 `void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( )   
[protected, inherited]`

Detach all singular iterators.

#### Postcondition

for all iterators *i* attached to this sequence, `i->_M_version == _M_version`.

5.249.2.7 `__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_  
base::_M_get_mutex ( ) throw () [protected,  
inherited]`

For use in [\\_Safe\\_sequence](#).

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if()`,  
and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_from_if()`.

5.249.2.8 `void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( )   
const [inline, inherited]`

Invalidates all iterators.

Definition at line 234 of file `safe_base.h`.

References `__gnu_debug::_Safe_sequence_base::_M_version`.

5.249.2.9 `void __gnu_debug::_Safe_sequence< multiset<_Key,_Compare,  
_Allocator>>::_M_invalidate_if ( _Predicate __pred )   
[inherited]`

Invalidates all iterators *x* that reference  
this sequence, are not singular, and for which `pred(x)` returns `true`. `pred` will be  
invoked with the normal iterators nested in the safe ones.

**5.249.2.10** void \_\_gnu\_debug::Safe\_sequence\_base::\_M\_revalidate\_singular ( ) [protected, inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

**5.249.2.11** void \_\_gnu\_debug::Safe\_sequence\_base::\_M\_swap ( \_Safe\_sequence\_base & \_\_x ) [protected, inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

**5.249.2.12** void \_\_gnu\_debug::Safe\_sequence< multiset< \_Key, \_Compare, \_Allocator > >::\_M\_transfer\_from\_if ( \_Safe\_sequence< multiset< \_Key, \_Compare, \_Allocator > > & \_\_from, \_Predicate \_\_pred ) [inherited]

Transfers all iterators *x* that reference *from* sequence, are not singular, and for which *pred(x)* returns *true*. *pred* will be invoked with the normal iterators nested in the safe ones.

### 5.249.3 Member Data Documentation

**5.249.3.1** \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_sequence\_base::\_M\_const\_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 185 of file *safe\_base.h*.

Referenced by \_\_gnu\_debug::Safe\_sequence< \_Sequence >::\_M\_invalidate\_if(), and \_\_gnu\_debug::Safe\_sequence< \_Sequence >::\_M\_transfer\_from\_if().

**5.249.3.2** \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_sequence\_base::\_M\_iterators [inherited]

The list of mutable iterators that reference this container.

## 5.250 `std::__debug::set< _Key, _Compare, _Allocator >` Class Template Reference

1445

Definition at line 182 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if()`.

### 5.249.3.3 `unsigned int __gnu_debug::_Safe_sequence_base::_M_version` [mutable, inherited]

The container version number. This number may never be 0.

Definition at line 188 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if()`.

The documentation for this class was generated from the following file:

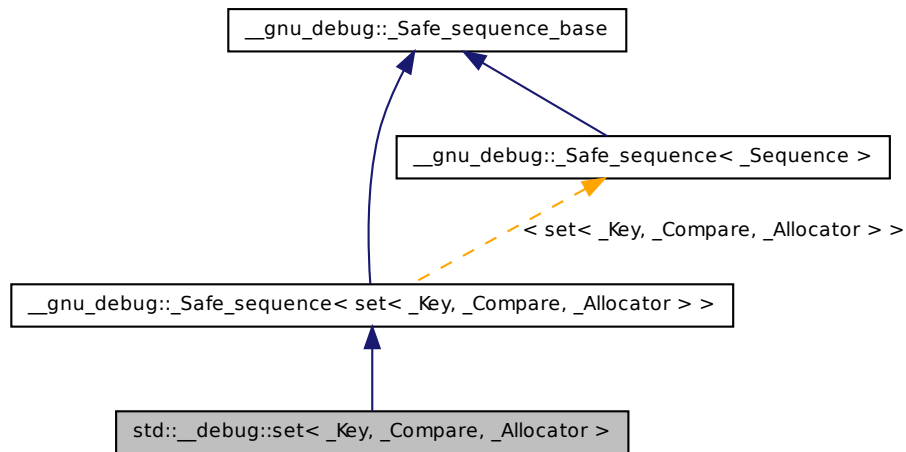
- [debug/multiset.h](#)

## 5.250 `std::__debug::set< _Key, _Compare, _Allocator >` Class Template Reference

Class [std::set](#) with safety/checking/debug instrumentation.



Inheritance diagram for `std::__debug::set<_Key, _Compare, _Allocator>`:



## Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::Safe_iterator<_Base_const_iterator, set>` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator<const_iterator>` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::Safe_iterator<_Base_iterator, set>` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator<iterator>` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Compare` **value\_compare**
- typedef `_Key` **value\_type**

### Public Member Functions

- `set` (`const _Compare &__comp=_Compare()`, `const _Allocator &__a=_Allocator()`)
- `template<typename _InputIterator>`  
`set` (`_InputIterator __first, _InputIterator __last, const _Compare &__comp=_Compare()`, `const _Allocator &__a=_Allocator()`)
- `set` (`const _Base &__x`)
- `set` (`set &&__x`)
- `set` (`const set &__x`)
- `set` (`initializer_list< value_type > __l, const _Compare &__comp=_Compare()`, `const allocator_type &__a=allocator_type()`)
- `void _M_attach` (`_Safe_iterator_base *__it, bool __constant`)
- `void _M_attach_single` (`_Safe_iterator_base *__it, bool __constant`) `throw ()`
- `_Base & _M_base` ()
- `const _Base & _M_base` () `const`
- `void _M_detach` (`_Safe_iterator_base *__it`)
- `void _M_detach_single` (`_Safe_iterator_base *__it`) `throw ()`
- `void _M_invalidate_all` () `const`
- `void _M_invalidate_if` (`_Predicate __pred`)
- `void _M_transfer_from_if` (`_Safe_sequence &__from, _Predicate __pred`)
- `const_iterator begin` () `const`
- `iterator begin` ()
- `const_iterator cbegin` () `const`
- `const_iterator cend` () `const`
- `void clear` ()
- `const_reverse_iterator crbegin` () `const`
- `const_reverse_iterator crend` () `const`
- `iterator end` ()
- `const_iterator end` () `const`
- `std::pair< iterator, iterator > equal_range` (`const key_type &__x`)
- `std::pair< const_iterator, const_iterator > equal_range` (`const key_type &__x`) `const`
- `iterator erase` (`const_iterator __first, const_iterator __last`)
- `iterator erase` (`const_iterator __position`)
- `size_type erase` (`const key_type &__x`)
- `iterator find` (`const key_type &__x`)
- `const_iterator find` (`const key_type &__x`) `const`
- `iterator insert` (`const_iterator __position, const value_type &__x`)
- `std::pair< iterator, bool > insert` (`const value_type &__x`)
- `std::pair< iterator, bool > insert` (`value_type &&__x`)
- `iterator insert` (`const_iterator __position, value_type &&__x`)
- `void insert` (`initializer_list< value_type > __l`)

- `template<typename _InputIterator>`  
  `void insert (_InputIterator __first, _InputIterator __last)`
- `iterator lower_bound (const key_type &__x)`
- `const_iterator lower_bound (const key_type &__x) const`
- `set & operator= (const set &__x)`
- `set & operator= (set &&__x)`
- `set & operator= (initializer_list< value_type > __l)`
- `reverse_iterator rbegin ()`
- `const_reverse_iterator rbegin () const`
- `reverse_iterator rend ()`
- `const_reverse_iterator rend () const`
- `void swap (set &__x)`
- `iterator upper_bound (const key_type &__x)`
- `const_iterator upper_bound (const key_type &__x) const`

#### Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

#### Protected Member Functions

- `void _M_detach_all ()`
- `void _M_detach_singular ()`
- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`
- `void _M_revalidate_singular ()`
- `void _M_swap (_Safe_sequence_base &__x)`

#### 5.250.1 Detailed Description

`template<typename _Key, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<_Key>> class std::__debug::set<_Key, _Compare, _Allocator>`

Class `std::set` with safety/checking/debug instrumentation.

Definition at line 44 of file `debug/set.h`.

## 5.250.2 Member Function Documentation

5.250.2.1 `void __gnu_debug::_Safe_sequence_base::_M_attach (`  
`_Safe_iterator_base * __it, bool __constant ) [inherited]`

Attach an iterator to this sequence.

5.250.2.2 `void __gnu_debug::_Safe_sequence_base::_M_attach_single`  
`( _Safe_iterator_base * __it, bool __constant ) throw ()`  
`[inherited]`

Likewise but not thread safe.

5.250.2.3 `void __gnu_debug::_Safe_sequence_base::_M_detach (`  
`_Safe_iterator_base * __it ) [inherited]`

Detach an iterator from this sequence

5.250.2.4 `void __gnu_debug::_Safe_sequence_base::_M_detach_all ( )`  
`[protected, inherited]`

Detach all iterators, leaving them singular.

Referenced by `__gnu_debug::_Safe_sequence_base::~~_Safe_sequence_base()`.

5.250.2.5 `void __gnu_debug::_Safe_sequence_base::_M_detach_single (`  
`_Safe_iterator_base * __it ) throw () [inherited]`

Likewise but not thread safe.

5.250.2.6 `void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( )`  
`[protected, inherited]`

Detach all singular iterators.

### Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

**5.250.2.7** `__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_  
base::_M_get_mutex ( ) throw () [protected,  
inherited]`

For use in [\\_Safe\\_sequence](#).

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`,  
and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if()`.

**5.250.2.8** `void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( )  
const [inline, inherited]`

Invalidates all iterators.

Definition at line 234 of file `safe_base.h`.

References `__gnu_debug::_Safe_sequence_base::_M_version`.

**5.250.2.9** `void __gnu_debug::_Safe_sequence< set< _Key, _Compare,  
_Allocator > >::_M_invalidate_if ( _Predicate __pred )  
[inherited]`

Invalidates all iterators `x` that reference  
this sequence, are not singular, and for which `pred(x)` returns `true`. `pred` will be  
invoked with the normal iterators nested in the safe ones.

**5.250.2.10** `void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( )  
[protected, inherited]`

Revalidates all attached singular iterators. This method may be used to validate  
iterators that were invalidated before (but for some reason, such as an exception, need  
to become valid again).

**5.250.2.11** `void __gnu_debug::_Safe_sequence_base::_M_swap ( )  
_Safe_sequence_base & __x ) [protected, inherited]`

Swap this sequence with the given sequence. This  
operation also swaps ownership of the iterators, so that when the operation is complete  
all iterators that originally referenced one container now reference the other container.

**5.250.2.12 void \_\_gnu\_debug::\_Safe\_sequence< set< \_Key, \_Compare, \_Allocator > >::\_M\_transfer\_from\_if ( \_Safe\_sequence< set< \_Key, \_Compare, \_Allocator > > & \_\_from, \_Predicate \_\_pred )  
[inherited]**

Transfers all iterators *x* that reference *from* sequence, are not singular, and for which *pred(x)* returns *true*. *pred* will be invoked with the normal iterators nested in the safe ones.

### **5.250.3 Member Data Documentation**

**5.250.3.1 \_Safe\_iterator\_base\* \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_const\_iterators [inherited]**

The list of constant iterators that reference this container.

Definition at line 185 of file *safe\_base.h*.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if()`.

**5.250.3.2 \_Safe\_iterator\_base\* \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_iterators [inherited]**

The list of mutable iterators that reference this container.

Definition at line 182 of file *safe\_base.h*.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if()`.

**5.250.3.3 unsigned int \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_version [mutable, inherited]**

The container version number. This number may never be 0.

Definition at line 188 of file *safe\_base.h*.

Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if()`.

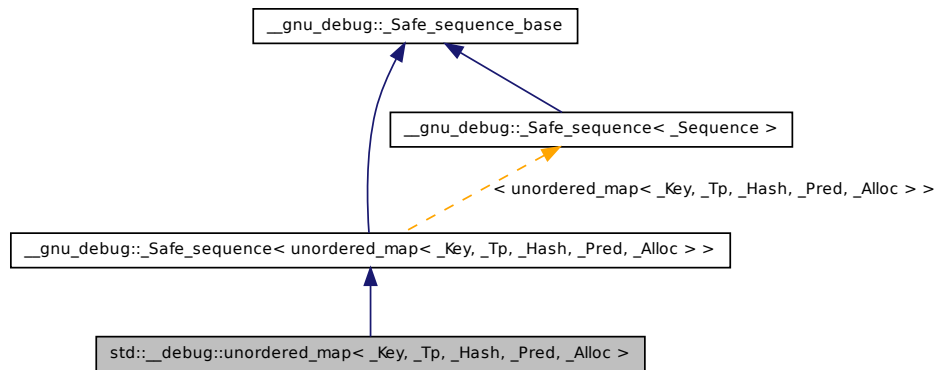
The documentation for this class was generated from the following file:

- [debug/set.h](#)

## 5.251 `std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >` Class Template Reference

Class `std::unordered_map` with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`:



### Public Types

- typedef `_Base::allocator_type` **allocator\_type**
- typedef `__gnu_debug::_Safe_iterator< _Base_const_iterator, unordered_map >` **const\_iterator**
- typedef `_Base::hasher` **hasher**
- typedef `__gnu_debug::_Safe_iterator< _Base_iterator, unordered_map >` **iterator**
- typedef `_Base::key_equal` **key\_equal**
- typedef `_Base::key_type` **key\_type**
- typedef `_Base::size_type` **size\_type**
- typedef `_Base::value_type` **value\_type**

### Public Member Functions

- **unordered\_map** (size\_type \_\_n=10, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())

- `template<typename _InputIterator>`  
`unordered_map` (`_InputIterator __first`, `_InputIterator __last`, `size_type __n=0`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- `unordered_map` (`const _Base &__x`)
- `unordered_map` (`unordered_map &&__x`)
- `unordered_map` (`const unordered_map &__x`)
- `unordered_map` (`initializer_list< value_type > __l`, `size_type __n=0`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- `void _M_attach` (`_Safe_iterator_base *__it`, `bool __constant`)
- `void _M_attach_single` (`_Safe_iterator_base *__it`, `bool __constant`) `throw ()`
- `const _Base & _M_base` (`() const`)
- `_Base & _M_base` (`()`)
- `void _M_detach` (`_Safe_iterator_base *__it`)
- `void _M_detach_single` (`_Safe_iterator_base *__it`) `throw ()`
- `void _M_invalidate_all` (`() const`)
- `void _M_invalidate_if` (`_Predicate __pred`)
- `void _M_transfer_from_if` (`_Safe_sequence &__from`, `_Predicate __pred`)
- `iterator begin` (`()`)
- `const_iterator begin` (`() const`)
- `const_iterator cbegin` (`() const`)
- `const_iterator cend` (`() const`)
- `void clear` (`()`)
- `const_iterator end` (`() const`)
- `iterator end` (`()`)
- `std::pair< const_iterator, const_iterator > equal_range` (`const key_type &__key`) `const`
- `std::pair< iterator, iterator > equal_range` (`const key_type &__key`)
- `iterator erase` (`const_iterator __first`, `const_iterator __last`)
- `iterator erase` (`const_iterator __it`)
- `size_type erase` (`const key_type &__key`)
- `iterator find` (`const key_type &__key`)
- `const_iterator find` (`const key_type &__key`) `const`
- `iterator insert` (`const_iterator __hint`, `const value_type &__obj`)
- `template<typename _Pair, typename = typename std::enable_if<std::is_convertible<_Pair, value_type>::value>::type>`  
`std::pair< iterator, bool > insert` (`_Pair &&__obj`)
- `template<typename _InputIterator>`  
`void insert` (`_InputIterator __first`, `_InputIterator __last`)
- `std::pair< iterator, bool > insert` (`const value_type &__obj`)
- `void insert` (`std::initializer_list< value_type > __l`)



- `template<typename _Pair, typename = typename std::enable_if<std::is_convertible<_Pair, value_type>::value>::type>`  
`iterator insert (const_iterator __hint, _Pair &&__obj)`
- `unordered_map & operator= (unordered_map &&__x)`
- `unordered_map & operator= (initializer_list< value_type > __l)`
- `unordered_map & operator= (const unordered_map &__x)`
- `void swap (unordered_map &__x)`

### Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

### Protected Member Functions

- `void _M_detach_all ()`
- `void _M_detach_singular ()`
- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`
- `void _M_revalidate_singular ()`
- `void _M_swap (_Safe_sequence_base &__x)`

#### 5.251.1 Detailed Description

`template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>,  
typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<_  
Key>> class std::__debug::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc  
>`

Class `std::unordered_map` with safety/checking/debug instrumentation.

Definition at line 50 of file `debug/unordered_map`.

#### 5.251.2 Member Function Documentation

5.251.2.1 `void __gnu_debug::Safe_sequence_base::_M_attach (  
_Safe_iterator_base * __it, bool __constant ) [inherited]`

Attach an iterator to this sequence.

5.251.2.2 void \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_attach\_single  
( \_Safe\_iterator\_base \* \_\_it, bool \_\_constant ) throw ()  
[inherited]

Likewise but not thread safe.

5.251.2.3 void \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_detach (   
\_Safe\_iterator\_base \* \_\_it ) [inherited]

Detach an iterator from this sequence

5.251.2.4 void \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_detach\_all ( )  
[protected, inherited]

Detach all iterators, leaving them singular.

Referenced by \_\_gnu\_debug::\_Safe\_sequence\_base::~~\_Safe\_sequence\_base().

5.251.2.5 void \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_detach\_single (   
\_Safe\_iterator\_base \* \_\_it ) throw () [inherited]

Likewise but not thread safe.

5.251.2.6 void \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_detach\_singular ( )  
[protected, inherited]

Detach all singular iterators.

#### Postcondition

for all iterators *i* attached to this sequence, *i*->\_M\_version == \_M\_version.

5.251.2.7 \_\_gnu\_cxx::\_\_mutex& \_\_gnu\_debug::\_Safe\_sequence\_  
base::\_M\_get\_mutex ( ) throw () [protected,  
inherited]

For use in [\\_Safe\\_sequence](#).

Referenced by \_\_gnu\_debug::\_Safe\_sequence< \_Sequence >::\_M\_invalidate\_if(),  
and \_\_gnu\_debug::\_Safe\_sequence< \_Sequence >::\_M\_transfer\_from\_if().

5.251.2.8 void \_\_gnu\_debug::Safe\_sequence\_base::\_M\_invalidate\_all ( )  
const [inline, inherited]

Invalidates all iterators.

Definition at line 234 of file safe\_base.h.

References \_\_gnu\_debug::Safe\_sequence\_base::\_M\_version.

5.251.2.9 void \_\_gnu\_debug::Safe\_sequence< unordered\_map<\_Key, \_Tp,  
\_Hash, \_Pred, \_Alloc > >::\_M\_invalidate\_if ( \_Predicate \_\_pred )  
[inherited]

Invalidates all iterators  $x$  that reference this sequence, are not singular, and for which `pred(x)` returns `true`. `pred` will be invoked with the normal iterators nested in the safe ones.

5.251.2.10 void \_\_gnu\_debug::Safe\_sequence\_base::\_M\_revalidate\_singular ( )  
[protected, inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

5.251.2.11 void \_\_gnu\_debug::Safe\_sequence\_base::\_M\_swap (   
\_Safe\_sequence\_base & \_\_x ) [protected, inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

5.251.2.12 void \_\_gnu\_debug::Safe\_sequence< unordered\_map<\_Key, \_Tp,  
\_Hash, \_Pred, \_Alloc > >::\_M\_transfer\_from\_if ( \_Safe\_sequence<  
unordered\_map<\_Key, \_Tp, \_Hash, \_Pred, \_Alloc > > & \_\_from,  
\_Predicate \_\_pred ) [inherited]

Transfers all iterators  $x$  that reference `from` sequence, are not singular, and for which `pred(x)` returns `true`. `pred` will be invoked with the normal iterators nested in the safe ones.

### 5.251.3 Member Data Documentation

#### 5.251.3.1 \_Safe\_iterator\_base\* \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_const\_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 185 of file safe\_base.h.

Referenced by \_\_gnu\_debug::\_Safe\_sequence< \_Sequence >::\_M\_invalidate\_if(), and \_\_gnu\_debug::\_Safe\_sequence< \_Sequence >::\_M\_transfer\_from\_if().

#### 5.251.3.2 \_Safe\_iterator\_base\* \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_iterators [inherited]

The list of mutable iterators that reference this container.

Definition at line 182 of file safe\_base.h.

Referenced by \_\_gnu\_debug::\_Safe\_sequence< \_Sequence >::\_M\_invalidate\_if(), and \_\_gnu\_debug::\_Safe\_sequence< \_Sequence >::\_M\_transfer\_from\_if().

#### 5.251.3.3 unsigned int \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_version [mutable, inherited]

The container version number. This number may never be 0.

Definition at line 188 of file safe\_base.h.

Referenced by \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_invalidate\_all(), and \_\_gnu\_debug::\_Safe\_sequence< \_Sequence >::\_M\_transfer\_from\_if().

The documentation for this class was generated from the following file:

- [debug/unordered\\_map](#)

### 5.252 std::\_\_debug::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc > Class Template Reference

Class [std::unordered\\_multimap](#) with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`:



## Public Types

- typedef `_Base::allocator_type` **allocator\_type**
- typedef `__gnu_debug::Safe_iterator< _Base_const_iterator, unordered_multimap >` **const\_iterator**
- typedef `_Base::hasher` **hasher**
- typedef `__gnu_debug::Safe_iterator< _Base_iterator, unordered_multimap >` **iterator**
- typedef `_Base::key_equal` **key\_equal**
- typedef `_Base::key_type` **key\_type**
- typedef `_Base::size_type` **size\_type**
- typedef `_Base::value_type` **value\_type**

## Public Member Functions

- **unordered\_multimap** (size\_type \_\_n=10, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator >  
**unordered\_multimap** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_multimap** (const `_Base` &\_\_x)
- **unordered\_multimap** (`unordered_multimap` &&\_\_x)
- **unordered\_multimap** (const `unordered_multimap` &\_\_x)
- **unordered\_multimap** (initializer\_list< value\_type > \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- void `_M_attach` (\_Safe\_iterator\_base \* \_\_it, bool \_\_constant)
- void `_M_attach_single` (\_Safe\_iterator\_base \* \_\_it, bool \_\_constant) throw ()
- const `_Base` & `_M_base` () const
- `_Base` & `_M_base` ()
- void `_M_detach` (\_Safe\_iterator\_base \* \_\_it)
- void `_M_detach_single` (\_Safe\_iterator\_base \* \_\_it) throw ()
- void `_M_invalidate_all` () const
- void `_M_invalidate_if` (\_Predicate \_\_pred)

- `void _M_transfer_from_if` (`_Safe_sequence &__from`, `_Predicate __pred`)
- `iterator begin` ()
- `const_iterator begin` () `const`
- `const_iterator cbegin` () `const`
- `const_iterator cend` () `const`
- `void clear` ()
- `const_iterator end` () `const`
- `iterator end` ()
- `std::pair< const_iterator, const_iterator > equal_range` (`const key_type &__key`) `const`
- `std::pair< iterator, iterator > equal_range` (`const key_type &__key`)
- `iterator erase` (`const_iterator __first`, `const_iterator __last`)
- `iterator erase` (`const_iterator __it`)
- `size_type erase` (`const key_type &__key`)
- `iterator find` (`const key_type &__key`)
- `const_iterator find` (`const key_type &__key`) `const`
- `iterator insert` (`const_iterator __hint`, `const value_type &__obj`)
- `template<typename _Pair, typename = typename std::enable_if<std::is_convertible<_Pair, value_type>::value>::type>`  
`iterator insert` (`_Pair &&__obj`)
- `template<typename _InputIterator >`  
`void insert` (`_InputIterator __first`, `_InputIterator __last`)
- `iterator insert` (`const value_type &__obj`)
- `void insert` (`std::initializer_list< value_type > __l`)
- `template<typename _Pair, typename = typename std::enable_if<std::is_convertible<_Pair, value_type>::value>::type>`  
`iterator insert` (`const_iterator __hint`, `_Pair &&__obj`)
- `unordered_multimap & operator=` (`unordered_multimap &&__x`)
- `unordered_multimap & operator=` (`initializer_list< value_type > __l`)
- `unordered_multimap & operator=` (`const unordered_multimap &__x`)
- `void swap` (`unordered_multimap &__x`)

#### Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

### Protected Member Functions

- `void _M_detach_all()`
- `void _M_detach_singular()`
- `__gnu_cxx::__mutex & _M_get_mutex() throw()`
- `void _M_revalidate_singular()`
- `void _M_swap(_Safe_sequence_base &__x)`

#### 5.252.1 Detailed Description

`template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>,  
typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<_  
_Key>> class std::__debug::unordered_multimap<_Key, _Tp, _Hash, _Pred, _  
Alloc>`

Class `std::unordered_multimap` with safety/checking/debug instrumentation.

Definition at line 337 of file `debug/unordered_map`.

#### 5.252.2 Member Function Documentation

5.252.2.1 `void __gnu_debug::Safe_sequence_base::_M_attach (  
_Safe_iterator_base * __it, bool __constant ) [inherited]`

Attach an iterator to this sequence.

5.252.2.2 `void __gnu_debug::Safe_sequence_base::_M_attach_single  
( _Safe_iterator_base * __it, bool __constant ) throw ()  
[inherited]`

Likewise but not thread safe.

5.252.2.3 `void __gnu_debug::Safe_sequence_base::_M_detach (  
_Safe_iterator_base * __it ) [inherited]`

Detach an iterator from this sequence

5.252.2.4 `void __gnu_debug::Safe_sequence_base::_M_detach_all ( )  
[protected, inherited]`

Detach all iterators, leaving them singular.

Referenced by `__gnu_debug::Safe_sequence_base::~~Safe_sequence_base()`.

5.252.2.5 `void __gnu_debug::_Safe_sequence_base::_M_detach_single (`  
`_Safe_iterator_base * __it ) throw () [inherited]`

Likewise but not thread safe.

5.252.2.6 `void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( )`  
`[protected, inherited]`

Detach all singular iterators.

#### Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

5.252.2.7 `__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_-`  
`base::_M_get_mutex ( ) throw () [protected,`  
`inherited]`

For use in [\\_Safe\\_sequence](#).

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if()`,  
and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_from_if()`.

5.252.2.8 `void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( )`  
`const [inline, inherited]`

Invalidates all iterators.

Definition at line 234 of file `safe_base.h`.

References `__gnu_debug::_Safe_sequence_base::_M_version`.

5.252.2.9 `void __gnu_debug::_Safe_sequence<unordered_multimap<_Key,`  
`_Tp, _Hash, _Pred, _Alloc>>::_M_invalidate_if ( _Predicate`  
`__pred ) [inherited]`

Invalidates all iterators `x` that reference  
this sequence, are not singular, and for which `pred(x)` returns `true`. `pred` will be  
invoked with the normal iterators nested in the safe ones.



**5.252.2.10** void \_\_gnu\_debug::Safe\_sequence\_base::\_M\_revalidate\_singular (  
 ) [protected, inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

**5.252.2.11** void \_\_gnu\_debug::Safe\_sequence\_base::\_M\_swap (  
 \_Safe\_sequence\_base & \_\_x ) [protected, inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

**5.252.2.12** void \_\_gnu\_debug::Safe\_sequence< unordered\_multimap<  
 \_Key, \_Tp, \_Hash, \_Pred, \_Alloc > >::\_M\_transfer\_from\_if (  
 \_Safe\_sequence< unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred,  
 \_Alloc > > & \_\_from, \_Predicate \_\_pred ) [inherited]

Transfers all iterators *x* that reference *from* sequence, are not singular, and for which *pred(x)* returns true. *pred* will be invoked with the normal iterators nested in the safe ones.

### 5.252.3 Member Data Documentation

**5.252.3.1** \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_sequence\_base::\_M\_  
const\_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 185 of file safe\_base.h.

Referenced by \_\_gnu\_debug::Safe\_sequence< \_Sequence >::\_M\_invalidate\_if(),  
and \_\_gnu\_debug::Safe\_sequence< \_Sequence >::\_M\_transfer\_from\_if().

**5.252.3.2** \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_sequence\_base::\_M\_  
iterators [inherited]

The list of mutable iterators that reference this container.

## 5.253 `std::__debug::unordered_multiset<_Value, _Hash, _Pred, _Alloc>` Class Template Reference 1463

Definition at line 182 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if()`, and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_from_if()`.

### 5.252.3.3 `unsigned int __gnu_debug::_Safe_sequence_base::_M_version` [mutable, inherited]

The container version number. This number may never be 0.

Definition at line 188 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`, and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_from_if()`.

The documentation for this class was generated from the following file:

- [debug/unordered\\_map](#)

## 5.253 `std::__debug::unordered_multiset<_Value, _Hash, _Pred, _Alloc>` Class Template Reference

Class `std::unordered_multiset` with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::unordered_multiset<_Value, _Hash, _Pred, _Alloc>`:



### Public Types

- typedef `_Base::allocator_type` **allocator\_type**
- typedef `__gnu_debug::_Safe_iterator<_Base_const_iterator, unordered_multiset>` **const\_iterator**
- typedef `_Base::hasher` **hasher**
- typedef `__gnu_debug::_Safe_iterator<_Base_iterator, unordered_multiset>` **iterator**
- typedef `_Base::key_equal` **key\_equal**
- typedef `_Base::key_type` **key\_type**
- typedef `_Base::size_type` **size\_type**
- typedef `_Base::value_type` **value\_type**

## Public Member Functions

- **unordered\_multiset** (size\_type \_\_n=10, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator >  
**unordered\_multiset** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_multiset** (const [\\_Base](#) &\_\_x)
- **unordered\_multiset** ([unordered\\_multiset](#) &&\_\_x)
- **unordered\_multiset** (const [unordered\\_multiset](#) &&\_\_x)
- **unordered\_multiset** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- void [\\_M\\_attach](#) (\_Safe\_iterator\_base \*\_\_it, bool \_\_constant)
- void [\\_M\\_attach\\_single](#) (\_Safe\_iterator\_base \*\_\_it, bool \_\_constant) throw ()
- const [\\_Base](#) & [\\_M\\_base](#) () const
- [\\_Base](#) & [\\_M\\_base](#) ()
- void [\\_M\\_detach](#) (\_Safe\_iterator\_base \*\_\_it)
- void [\\_M\\_detach\\_single](#) (\_Safe\_iterator\_base \*\_\_it) throw ()
- void [\\_M\\_invalidate\\_all](#) () const
- void [\\_M\\_invalidate\\_if](#) (\_Predicate \_\_pred)
- void [\\_M\\_transfer\\_from\\_if](#) (\_Safe\_sequence &\_\_from, \_Predicate \_\_pred)
- [iterator](#) **begin** ()
- const [iterator](#) **begin** () const
- const [iterator](#) **cbegin** () const
- const [iterator](#) **cend** () const
- void **clear** ()
- const [iterator](#) **end** () const
- [iterator](#) **end** ()
- [std::pair](#)< [const\\_iterator](#), [const\\_iterator](#) > **equal\_range** (const key\_type &\_\_key) const
- [std::pair](#)< [iterator](#), [iterator](#) > **equal\_range** (const key\_type &\_\_key)
- [iterator](#) **erase** ([const\\_iterator](#) \_\_first, [const\\_iterator](#) \_\_last)
- [iterator](#) **erase** ([const\\_iterator](#) \_\_it)
- size\_type **erase** (const key\_type &\_\_key)
- [iterator](#) **find** (const key\_type &\_\_key)
- const [iterator](#) **find** (const key\_type &\_\_key) const
- [iterator](#) **insert** ([const\\_iterator](#) \_\_hint, const value\_type &\_\_obj)
- [iterator](#) **insert** (value\_type &&\_\_obj)
- template<typename \_InputIterator >  
 void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- [iterator](#) **insert** (const value\_type &\_\_obj)

- `void insert (std::initializer_list< value_type > __l)`
- `iterator insert (const_iterator __hint, value_type &&__obj)`
- `unordered_multiset & operator= (unordered_multiset &&__x)`
- `unordered_multiset & operator= (initializer_list< value_type > __l)`
- `unordered_multiset & operator= (const unordered_multiset &__x)`
- `void swap (unordered_multiset &__x)`

### Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

### Protected Member Functions

- `void _M_detach_all ()`
- `void _M_detach_singular ()`
- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`
- `void _M_revalidate_singular ()`
- `void _M_swap (_Safe_sequence_base &__x)`

#### 5.253.1 Detailed Description

`template<typename _Value, typename _Hash = std::hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>>`  
**class** `std::__debug::unordered_multiset<_Value, _Hash, _Pred, _Alloc>`

Class `std::unordered_multiset` with safety/checking/debug instrumentation.

Definition at line 327 of file `debug/unordered_set`.

#### 5.253.2 Member Function Documentation

**5.253.2.1** `void __gnu_debug::_Safe_sequence_base::_M_attach (`  
`_Safe_iterator_base * __it, bool __constant ) [inherited]`

Attach an iterator to this sequence.

**5.253.2.2** `void __gnu_debug::_Safe_sequence_base::_M_attach_single`  
`( _Safe_iterator_base * __it, bool __constant ) throw ()`  
**[inherited]**

Likewise but not thread safe.

5.253.2.3 void \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_detach (   
\_Safe\_iterator\_base \* \_\_it ) [inherited]

Detach an iterator from this sequence

5.253.2.4 void \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_detach\_all ( )   
[protected, inherited]

Detach all iterators, leaving them singular.

Referenced by \_\_gnu\_debug::\_Safe\_sequence\_base::~~\_Safe\_sequence\_base().

5.253.2.5 void \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_detach\_single (   
\_Safe\_iterator\_base \* \_\_it ) throw () [inherited]

Likewise but not thread safe.

5.253.2.6 void \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_detach\_singular ( )   
[protected, inherited]

Detach all singular iterators.

#### Postcondition

for all iterators i attached to this sequence, i->\_M\_version == \_M\_version.

5.253.2.7 \_\_gnu\_cxx::mutex& \_\_gnu\_debug::\_Safe\_sequence\_  
base::\_M\_get\_mutex ( ) throw () [protected,  
inherited]

For use in [\\_Safe\\_sequence](#).

Referenced by \_\_gnu\_debug::\_Safe\_sequence< \_Sequence >::\_M\_invalidate\_if(),  
and \_\_gnu\_debug::\_Safe\_sequence< \_Sequence >::\_M\_transfer\_from\_if().

5.253.2.8 void \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_invalidate\_all ( )   
const [inline, inherited]

Invalidates all iterators.

Definition at line 234 of file safe\_base.h.

References \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_version.

**5.253.2.9** `void __gnu_debug::Safe_sequence< unordered_multiset< _Value, _Hash, _Pred, _Alloc > >::_M_invalidate_if ( _Predicate __pred )`  
**[inherited]**

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `pred(x)` returns `true`. `pred` will be invoked with the normal iterators nested in the safe ones.

**5.253.2.10** `void __gnu_debug::Safe_sequence_base::_M_revalidate_singular ( )` **[protected, inherited]**

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

**5.253.2.11** `void __gnu_debug::Safe_sequence_base::_M_swap ( _Safe_sequence_base & __x )` **[protected, inherited]**

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

**5.253.2.12** `void __gnu_debug::Safe_sequence< unordered_multiset< _Value, _Hash, _Pred, _Alloc > >::_M_transfer_from_if ( _Safe_sequence< unordered_multiset< _Value, _Hash, _Pred, _Alloc > > & __from, _Predicate __pred )` **[inherited]**

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `pred(x)` returns `true`. `pred` will be invoked with the normal iterators nested in the safe ones.

### 5.253.3 Member Data Documentation

**5.253.3.1** `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::_M_const_iterators` **[inherited]**

The list of constant iterators that reference this container.

Definition at line 185 of file `safe_base.h`.

## 5.254 `std::__debug::unordered_set<_Value, _Hash, _Pred, _Alloc>` Class Template Reference 1468

---

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if()`, and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_from_if()`.

### 5.253.3.2 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators` [inherited]

The list of mutable iterators that reference this container.

Definition at line 182 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`, and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_from_if()`.

### 5.253.3.3 `unsigned int __gnu_debug::_Safe_sequence_base::_M_version` [mutable, inherited]

The container version number. This number may never be 0.

Definition at line 188 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`, and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_from_if()`.

The documentation for this class was generated from the following file:

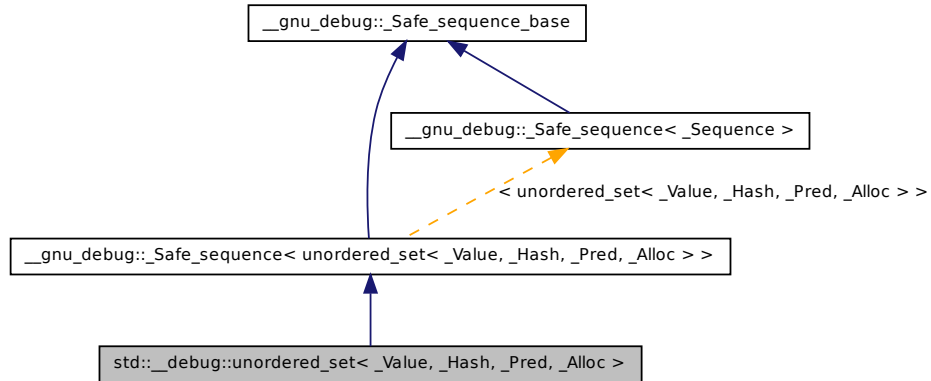
- [debug/unordered\\_set](#)

## 5.254 `std::__debug::unordered_set<_Value, _Hash, _Pred, _Alloc>` Class Template Reference

Class [std::unordered\\_set](#) with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::unordered_set<_Value, _Hash, _Pred, _Alloc>`

>:



## Public Types

- typedef `_Base::allocator_type` **allocator\_type**
- typedef `__gnu_debug::Safe_iterator< _Base_const_iterator, unordered_set >` **const\_iterator**
- typedef `_Base::hasher` **hasher**
- typedef `__gnu_debug::Safe_iterator< _Base_iterator, unordered_set >` **iterator**
- typedef `_Base::key_equal` **key\_equal**
- typedef `_Base::key_type` **key\_type**
- typedef `_Base::size_type` **size\_type**
- typedef `_Base::value_type` **value\_type**

## Public Member Functions

- **unordered\_set** (`size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- `template<typename _InputIterator>`  
**unordered\_set** (`_InputIterator __first`, `_InputIterator __last`, `size_type __n=0`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- **unordered\_set** (`const _Base &__x`)
- **unordered\_set** (`unordered_set &&__x`)



- `unordered_set` (const `unordered_set` &\_\_x)
- `unordered_set` (`initializer_list`< value\_type > \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- void `_M_attach` (\_Safe\_iterator\_base \* \_\_it, bool \_\_constant)
- void `_M_attach_single` (\_Safe\_iterator\_base \* \_\_it, bool \_\_constant) throw ()
- const `_Base` & `_M_base` () const
- `_Base` & `_M_base` ()
- void `_M_detach` (\_Safe\_iterator\_base \* \_\_it)
- void `_M_detach_single` (\_Safe\_iterator\_base \* \_\_it) throw ()
- void `_M_invalidate_all` () const
- void `_M_invalidate_if` (\_Predicate \_\_pred)
- void `_M_transfer_from_if` (\_Safe\_sequence & \_\_from, \_Predicate \_\_pred)
- `iterator begin` ()
- `const_iterator begin` () const
- `const_iterator cbegin` () const
- `const_iterator cend` () const
- void `clear` ()
- `const_iterator end` () const
- `iterator end` ()
- `std::pair`< `const_iterator`, `const_iterator` > `equal_range` (const key\_type & \_\_key) const
- `std::pair`< `iterator`, `iterator` > `equal_range` (const key\_type & \_\_key)
- `iterator erase` (`const_iterator` \_\_first, `const_iterator` \_\_last)
- `iterator erase` (`const_iterator` \_\_it)
- size\_type `erase` (const key\_type & \_\_key)
- `iterator find` (const key\_type & \_\_key)
- `const_iterator find` (const key\_type & \_\_key) const
- `iterator insert` (`const_iterator` \_\_hint, const value\_type & \_\_obj)
- `std::pair`< `iterator`, bool > `insert` (value\_type && \_\_obj)
- template<typename \_InputIterator>  
 void `insert` (\_InputIterator \_\_first, \_InputIterator \_\_last)
- `std::pair`< `iterator`, bool > `insert` (const value\_type & \_\_obj)
- void `insert` (`std::initializer_list`< value\_type > \_\_l)
- `iterator insert` (`const_iterator` \_\_hint, value\_type && \_\_obj)
- `unordered_set` & `operator=` (`unordered_set` && \_\_x)
- `unordered_set` & `operator=` (`initializer_list`< value\_type > \_\_l)
- `unordered_set` & `operator=` (const `unordered_set` & \_\_x)
- void `swap` (`unordered_set` & \_\_x)

## Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

## Protected Member Functions

- `void _M_detach_all ()`
- `void _M_detach_singular ()`
- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`
- `void _M_revalidate_singular ()`
- `void _M_swap (_Safe_sequence_base & __x)`

### 5.254.1 Detailed Description

```
template<typename _Value, typename _Hash = std::hash<_Value>, typename
_Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>>
class std::__debug::unordered_set<_Value, _Hash, _Pred, _Alloc>
```

Class `std::unordered_set` with safety/checking/debug instrumentation.

Definition at line 50 of file `debug/unordered_set`.

### 5.254.2 Member Function Documentation

**5.254.2.1** `void __gnu_debug::Safe_sequence_base::_M_attach (`  
`_Safe_iterator_base * __it, bool __constant ) [inherited]`

Attach an iterator to this sequence.

**5.254.2.2** `void __gnu_debug::Safe_sequence_base::_M_attach_single`  
`( _Safe_iterator_base * __it, bool __constant ) throw ()`  
`[inherited]`

Likewise but not thread safe.

**5.254.2.3** `void __gnu_debug::Safe_sequence_base::_M_detach (`  
`_Safe_iterator_base * __it ) [inherited]`

Detach an iterator from this sequence

5.254.2.4 `void __gnu_debug::Safe_sequence_base::M_detach_all ( )`  
[protected, inherited]

Detach all iterators, leaving them singular.

Referenced by `__gnu_debug::Safe_sequence_base::~~Safe_sequence_base()`.

5.254.2.5 `void __gnu_debug::Safe_sequence_base::M_detach_single (`  
`_Safe_iterator_base * __it ) throw ()` [inherited]

Likewise but not thread safe.

5.254.2.6 `void __gnu_debug::Safe_sequence_base::M_detach_singular ( )`  
[protected, inherited]

Detach all singular iterators.

#### Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

5.254.2.7 `__gnu_cxx::mutex& __gnu_debug::Safe_sequence_-`  
`base::M_get_mutex ( ) throw ()` [protected,  
inherited]

For use in [\\_Safe\\_sequence](#).

Referenced by `__gnu_debug::Safe_sequence<_Sequence>::_M_invalidate_if()`,  
and `__gnu_debug::Safe_sequence<_Sequence>::_M_transfer_from_if()`.

5.254.2.8 `void __gnu_debug::Safe_sequence_base::M_invalidate_all ( )`  
`const` [inline, inherited]

Invalidates all iterators.

Definition at line 234 of file `safe_base.h`.

References `__gnu_debug::Safe_sequence_base::_M_version`.

5.254.2.9 `void __gnu_debug::Safe_sequence<unordered_set<_Value,`  
`_Hash, _Pred, _Alloc>>::_M_invalidate_if ( _Predicate __pred )`  
[inherited]

Invalidates all iterators `x` that reference  
this sequence, are not singular, and for which `pred(x)` returns `true`. `pred` will be

invoked with the normal iterators nested in the safe ones.

**5.254.2.10** void \_\_gnu\_debug::Safe\_sequence\_base::M\_revalidate\_singular ( ) [protected, inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

**5.254.2.11** void \_\_gnu\_debug::Safe\_sequence\_base::M\_swap ( \_Safe\_sequence\_base & \_\_x ) [protected, inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

**5.254.2.12** void \_\_gnu\_debug::Safe\_sequence< unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc > >::M\_transfer\_from\_if ( \_Safe\_sequence< unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc > > & \_\_from, \_Predicate \_\_pred ) [inherited]

Transfers all iterators *x* that reference *from* sequence, are not singular, and for which *pred*(*x*) returns *true*. *pred* will be invoked with the normal iterators nested in the safe ones.

### 5.254.3 Member Data Documentation

**5.254.3.1** \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_sequence\_base::M\_const\_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 185 of file *safe\_base.h*.

Referenced by \_\_gnu\_debug::Safe\_sequence< \_Sequence >::M\_invalidate\_if(), and \_\_gnu\_debug::Safe\_sequence< \_Sequence >::M\_transfer\_from\_if().

**5.254.3.2** \_Safe\_iterator\_base\* \_\_gnu\_debug::Safe\_sequence\_base::M\_iterators [inherited]

## **5.255 `std::__debug::vector<_Tp, _Allocator>` Class Template Reference 1474**

The list of mutable iterators that reference this container.

Definition at line 182 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if()`, and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_from_if()`.

### **5.254.3.3 `unsigned int __gnu_debug::_Safe_sequence_base::_M_version` [mutable, inherited]**

The container version number. This number may never be 0.

Definition at line 188 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`, and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_from_if()`.

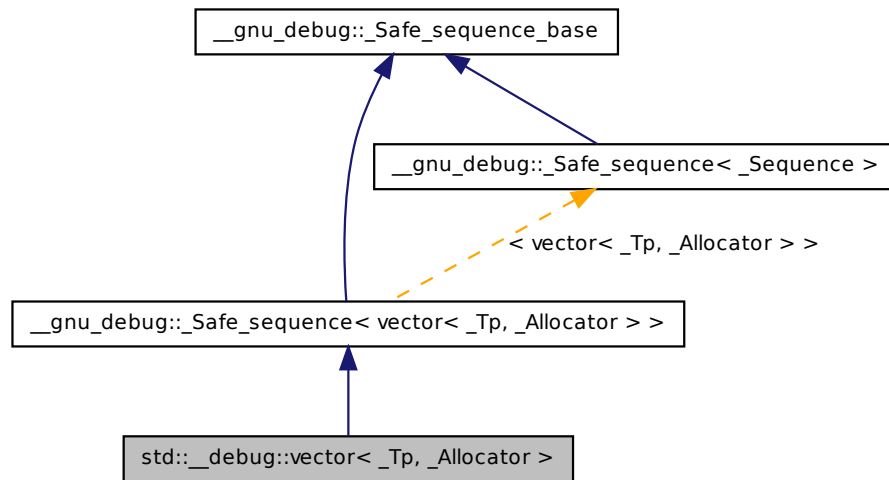
The documentation for this class was generated from the following file:

- [debug/unordered\\_set](#)

## **5.255 `std::__debug::vector<_Tp, _Allocator>` Class Template Reference**

Class [std::vector](#) with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::vector< _Tp, _Allocator >`:



### Public Types

- `typedef _Allocator allocator_type`
- `typedef __gnu_debug::Safe_iterator< _Base_const_iterator, vector > const_iterator`
- `typedef _Base::const_pointer const_pointer`
- `typedef _Base::const_reference const_reference`
- `typedef std::reverse_iterator< const_iterator > const_reverse_iterator`
- `typedef _Base::difference_type difference_type`
- `typedef __gnu_debug::Safe_iterator< _Base_iterator, vector > iterator`
- `typedef _Base::pointer pointer`
- `typedef _Base::reference reference`
- `typedef std::reverse_iterator< iterator > reverse_iterator`
- `typedef _Base::size_type size_type`
- `typedef _Tp value_type`

### Public Member Functions

- `vector (const _Allocator &__a=_Allocator())`

- **vector** (size\_type \_\_n)
- template<class \_InputIterator >  
**vector** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Allocator &\_\_a=\_Allocator())
- **vector** (initializer\_list< value\_type > \_\_l, const allocator\_type &\_\_a=allocator\_type())
- **vector** (const **vector** &\_\_x)
- **vector** (size\_type \_\_n, const \_Tp &\_\_value, const \_Allocator &\_\_a=\_Allocator())
- **vector** (const \_Base &\_\_x)
- **vector** (**vector** &&\_\_x)
- void **\_M\_attach** (\_Safe\_iterator\_base \*\_\_it, bool \_\_constant)
- void **\_M\_attach\_single** (\_Safe\_iterator\_base \*\_\_it, bool \_\_constant) throw ()
- **\_Base** & **\_M\_base** ()
- const **\_Base** & **\_M\_base** () const
- void **\_M\_detach** (\_Safe\_iterator\_base \*\_\_it)
- void **\_M\_detach\_single** (\_Safe\_iterator\_base \*\_\_it) throw ()
- void **\_M\_invalidate\_all** () const
- void **\_M\_invalidate\_if** (\_Predicate \_\_pred)
- void **\_M\_transfer\_from\_if** (\_Safe\_sequence &\_\_from, \_Predicate \_\_pred)
- void **assign** (size\_type \_\_n, const \_Tp &\_\_u)
- template<typename \_InputIterator >  
void **assign** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void **assign** (initializer\_list< value\_type > \_\_l)
- reference **back** ()
- const\_reference **back** () const
- **iterator** **begin** ()
- const\_iterator **begin** () const
- size\_type **capacity** () const
- const\_iterator **cbegin** () const
- const\_iterator **end** () const
- void **clear** ()
- const\_reverse\_iterator **crbegin** () const
- const\_reverse\_iterator **crend** () const
- template<typename... \_Args>  
**iterator** **emplace** (**iterator** \_\_position, \_Args &&...\_\_args)
- template<typename... \_Args>  
void **emplace\_back** (\_Args &&...\_\_args)
- **iterator** **end** ()
- const\_iterator **end** () const
- **iterator** **erase** (**iterator** \_\_position)
- **iterator** **erase** (**iterator** \_\_first, **iterator** \_\_last)
- reference **front** ()

- `const_reference front () const`
- `void insert (iterator __position, initializer_list< value_type > __l)`
- `void insert (iterator __position, size_type __n, const _Tp &__x)`
- `iterator insert (iterator __position, const _Tp &__x)`
- `template<typename _Up = _Tp>  
__gnu_cxx::__enable_if<!std::__are_same< _Up, bool >::__value, iterator  
>::__type insert (iterator __position, _Tp &&__x)`
- `template<class _InputIterator >  
void insert (iterator __position, _InputIterator __first, _InputIterator __last)`
- `vector & operator= (initializer_list< value_type > __l)`
- `vector & operator= (const vector &__x)`
- `vector & operator= (vector &&__x)`
- `reference operator[] (size_type __n)`
- `const_reference operator[] (size_type __n) const`
- `void pop_back ()`
- `template<typename _Up = _Tp>  
__gnu_cxx::__enable_if<!std::__are_same< _Up, bool >::__value, void >::__-  
_type push_back (_Tp &&__x)`
- `void push_back (const _Tp &__x)`
- `const_reverse_iterator rbegin () const`
- `reverse_iterator rbegin ()`
- `const_reverse_iterator rend () const`
- `reverse_iterator rend ()`
- `void reserve (size_type __n)`
- `void resize (size_type __sz, const _Tp &__c)`
- `void resize (size_type __sz)`
- `void swap (vector &__x)`

#### Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

#### Protected Member Functions

- `void _M_detach_all ()`
- `void _M_detach_singular ()`
- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`
- `void _M_revalidate_singular ()`
- `void _M_swap (_Safe_sequence_base &__x)`



## 5.255 `std::__debug::vector<_Tp, _Allocator>` > Class Template Reference 1478

### 5.255.1 Detailed Description

`template<typename _Tp, typename _Allocator = std::allocator<_Tp>> class std::__debug::vector<_Tp, _Allocator>`

Class [std::vector](#) with safety/checking/debug instrumentation.

Definition at line 45 of file `debug/vector`.

### 5.255.2 Constructor & Destructor Documentation

**5.255.2.1** `template<typename _Tp, typename _Allocator = std::allocator<_Tp>> std::__debug::vector<_Tp, _Allocator>::vector ( const _Base & __x ) [inline]`

Construction from a release-mode vector.

Definition at line 108 of file `debug/vector`.

### 5.255.3 Member Function Documentation

**5.255.3.1** `void __gnu_debug::_Safe_sequence_base::_M_attach ( _Safe_iterator_base * __it, bool __constant ) [inherited]`

Attach an iterator to this sequence.

**5.255.3.2** `void __gnu_debug::_Safe_sequence_base::_M_attach_single ( _Safe_iterator_base * __it, bool __constant ) throw () [inherited]`

Likewise but not thread safe.

**5.255.3.3** `void __gnu_debug::_Safe_sequence_base::_M_detach ( _Safe_iterator_base * __it ) [inherited]`

Detach an iterator from this sequence

**5.255.3.4** `void __gnu_debug::_Safe_sequence_base::_M_detach_all ( ) [protected, inherited]`

Detach all iterators, leaving them singular.

## 5.255 `std::__debug::vector<_Tp, _Allocator>` Class Template Reference 1479

Referenced by `__gnu_debug::_Safe_sequence_base::~~_Safe_sequence_base()`.

**5.255.3.5** `void __gnu_debug::_Safe_sequence_base::_M_detach_single (`  
`_Safe_iterator_base * __it ) throw () [inherited]`

Likewise but not thread safe.

**5.255.3.6** `void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( )`  
`[protected, inherited]`

Detach all singular iterators.

### Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

**5.255.3.7** `__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_-`  
`base::_M_get_mutex ( ) throw () [protected,`  
`inherited]`

For use in [\\_Safe\\_sequence](#).

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if()`,  
and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_from_if()`.

**5.255.3.8** `void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( )`  
`const [inline, inherited]`

Invalidates all iterators.

Definition at line 234 of file `safe_base.h`.

References `__gnu_debug::_Safe_sequence_base::_M_version`.

**5.255.3.9** `void __gnu_debug::_Safe_sequence<vector<_Tp, _Allocator>`  
`>::_M_invalidate_if ( _Predicate __pred ) [inherited]`

Invalidates all iterators `x` that reference  
this sequence, are not singular, and for which `pred(x)` returns `true`. `pred` will be  
invoked with the normal iterators nested in the safe ones.

## 5.255 `std::__debug::vector<_Tp, _Allocator>` Class Template Reference 1480

---

**5.255.3.10** `void __gnu_debug::Safe_sequence_base::_M_revalidate_singular ( ) [protected, inherited]`

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

**5.255.3.11** `void __gnu_debug::Safe_sequence_base::_M_swap ( _Safe_sequence_base & __x ) [protected, inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

**5.255.3.12** `void __gnu_debug::Safe_sequence< vector<_Tp, _Allocator> >::_M_transfer_from_if ( _Safe_sequence< vector<_Tp, _Allocator> > & __from, _Predicate __pred ) [inherited]`

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `pred(x)` returns `true`. `pred` will be invoked with the normal iterators nested in the safe ones.

### 5.255.4 Member Data Documentation

**5.255.4.1** `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::_M_const_iterators [inherited]`

The list of constant iterators that reference this container.

Definition at line 185 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::_M_invalidate_if()`, and `__gnu_debug::Safe_sequence< _Sequence >::_M_transfer_from_if()`.

**5.255.4.2** `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::_M_iterators [inherited]`

The list of mutable iterators that reference this container.

Definition at line 182 of file `safe_base.h`.

## 5.256 `std::__declval_protector<_Tp>` Struct Template Reference 1481

---

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if()`, and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_from_if()`.

### 5.255.4.3 `unsigned int __gnu_debug::_Safe_sequence_base::_M_version` [mutable, inherited]

The container version number. This number may never be 0.

Definition at line 188 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`, and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_from_if()`.

The documentation for this class was generated from the following file:

- [debug/vector](#)

## 5.256 `std::__declval_protector<_Tp>` Struct Template Reference

`declval`

### Static Public Member Functions

- static [add\\_rvalue\\_reference](#)<\_Tp>::type `__delegate()`

### Static Public Attributes

- static const bool `__stop`

### 5.256.1 Detailed Description

`template<typename _Tp> struct std::__declval_protector<_Tp>`

`declval`

Definition at line 1124 of file `type_traits`.

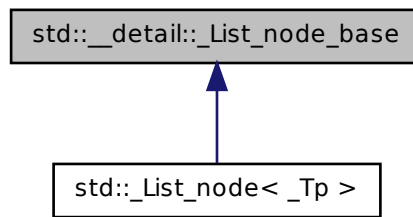
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.257 std::\_\_detail::\_List\_node\_base Struct Reference

Common part of a node in the list.

Inheritance diagram for std::\_\_detail::\_List\_node\_base:



### Public Member Functions

- void **\_M\_hook** ([\\_List\\_node\\_base](#) \*const \_\_position) throw ()
- void **\_M\_reverse** () throw ()
- void **\_M\_transfer** ([\\_List\\_node\\_base](#) \*const \_\_first, [\\_List\\_node\\_base](#) \*const \_\_last) throw ()
- void **\_M\_unhook** () throw ()

### Static Public Member Functions

- static void **swap** ([\\_List\\_node\\_base](#) &\_\_x, [\\_List\\_node\\_base](#) &\_\_y) throw ()

### Public Attributes

- [\\_List\\_node\\_base](#) \* **\_M\_next**
- [\\_List\\_node\\_base](#) \* **\_M\_prev**

#### 5.257.1 Detailed Description

Common part of a node in the list.

Definition at line 76 of file `stl_list.h`.

The documentation for this struct was generated from the following file:

- [stl\\_list.h](#)

## 5.258 `std::__exception_ptr::exception_ptr` Class Reference

An opaque pointer to an arbitrary exception.

### Public Member Functions

- `exception_ptr` (const `exception_ptr` &) throw ()
- `exception_ptr` (`exception_ptr` &&\_\_o) throw ()
- `exception_ptr` (nullptr\_t) throw ()
- const `type_info` \* `__cxa_exception_type` () const `__attribute__((__pure__))` throw ()
- `operator bool` () const
- `exception_ptr` & `operator=` (const `exception_ptr` &) throw ()
- `exception_ptr` & `operator=` (`exception_ptr` &&\_\_o) throw ()
- void `swap` (`exception_ptr` &) throw ()

### Friends

- bool `operator==` (const `exception_ptr` &, const `exception_ptr` &) `__attribute__((__pure__))` throw ()
- `exception_ptr` `std::current_exception` () throw ()
- void `std::rethrow_exception` (`exception_ptr`)

#### 5.258.1 Detailed Description

An opaque pointer to an arbitrary exception.

Definition at line 73 of file `exception_ptr.h`.

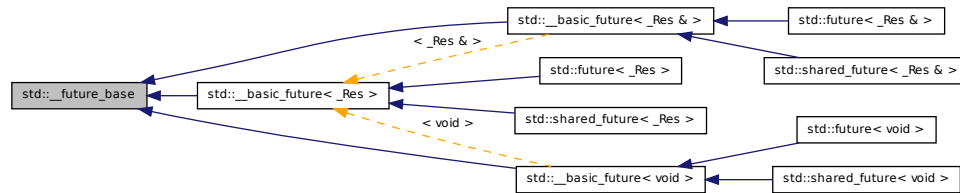
The documentation for this class was generated from the following file:

- [exception\\_ptr.h](#)

## 5.259 `std::__future_base` Struct Reference

Base class and enclosing scope.

Inheritance diagram for std::\_\_future\_base:



## Classes

- struct [\\_Ptr](#)  
*A [unique\\_ptr](#) based on the instantiating type.*
- struct [\\_Result](#)  
*Result.*
- struct [\\_Result<\\_Res &>](#)  
*Partial specialization for reference types.*
- struct [\\_Result<void>](#)  
*Explicit specialization for void.*
- struct [\\_Result\\_alloc](#)  
*Result\_alloc.*
- struct [\\_Result\\_base](#)  
*Base class for results.*
- class [\\_State](#)  
*Shared state between a promise and one or more associated futures.*

## Static Public Member Functions

- template<typename \_Res, typename \_Allocator>  
static [\\_Ptr<\\_Result\\_alloc<\\_Res, \\_Allocator>>](#)::type [\\_S\\_allocate\\_result](#)  
(const \_Allocator &\_\_a)

### 5.259.1 Detailed Description

Base class and enclosing scope.

Definition at line 156 of file future.

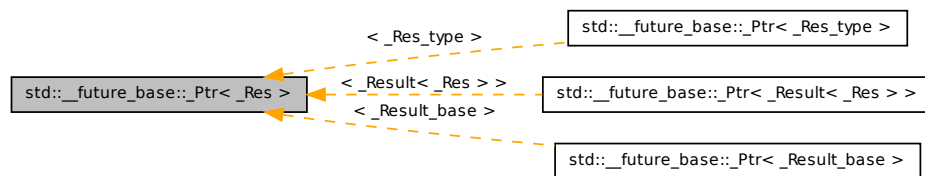
The documentation for this struct was generated from the following file:

- [future](#)

## 5.260 `std::__future_base::_Ptr<_Res>` Struct Template Reference

A [unique\\_ptr](#) based on the instantiating type.

Inheritance diagram for `std::__future_base::_Ptr<_Res>`:



### Public Types

- typedef [unique\\_ptr](#)<\_Res, \_Result\_base::\_Deleter> **type**

### 5.260.1 Detailed Description

`template<typename _Res> struct std::__future_base::_Ptr<_Res>`

A [unique\\_ptr](#) based on the instantiating type.

Definition at line 231 of file future.

The documentation for this struct was generated from the following file:

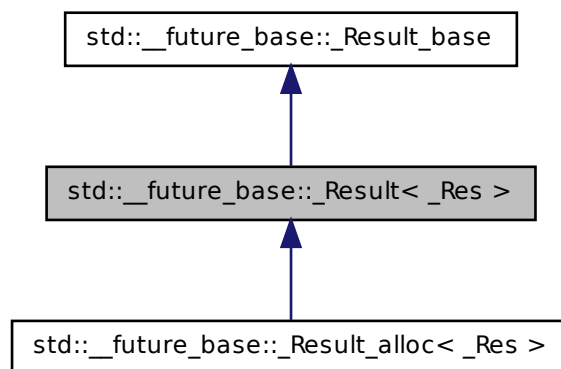
- [future](#)



## 5.261 `std::__future_base::_Result< _Res >` Struct Template Reference

Result.

Inheritance diagram for `std::__future_base::_Result< _Res >`:



### Public Member Functions

- `void _M_set (const _Res &__res)`
- `void _M_set (_Res &&__res)`
- `_Res & _M_value ()`

### Public Attributes

- `exception_ptr _M_error`

#### 5.261.1 Detailed Description

`template<typename _Res> struct std::__future_base::_Result< _Res >`

Result.

Definition at line 181 of file `future`.

## 5.262 `std::__future_base::_Result< _Res & >` Struct Template Reference 1487

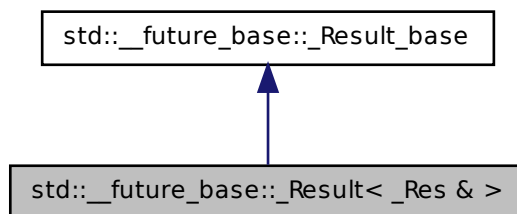
The documentation for this struct was generated from the following file:

- [future](#)

### 5.262 `std::__future_base::_Result< _Res & >` Struct Template Reference

Partial specialization for reference types.

Inheritance diagram for `std::__future_base::_Result< _Res & >`:



#### Public Member Functions

- `_Res & _M_get ()`
- `void _M_set (_Res &__res)`

#### Public Attributes

- `exception_ptr _M_error`

#### 5.262.1 Detailed Description

**template<typename \_Res> struct `std::__future_base::_Result< _Res & >`**

Partial specialization for reference types.

Definition at line 470 of file `future`.

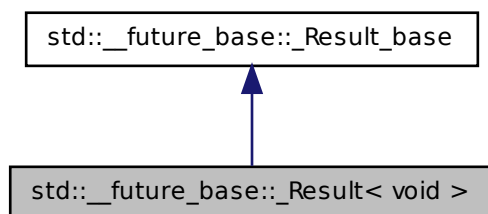
The documentation for this struct was generated from the following file:

- [future](#)

### 5.263 `std::__future_base::_Result< void >` Struct Template Reference

Explicit specialization for void.

Inheritance diagram for `std::__future_base::_Result< void >`:



#### Public Attributes

- `exception_ptr _M_error`

#### 5.263.1 Detailed Description

**template<> struct `std::__future_base::_Result< void >`**

Explicit specialization for void.

Definition at line 486 of file `future`.

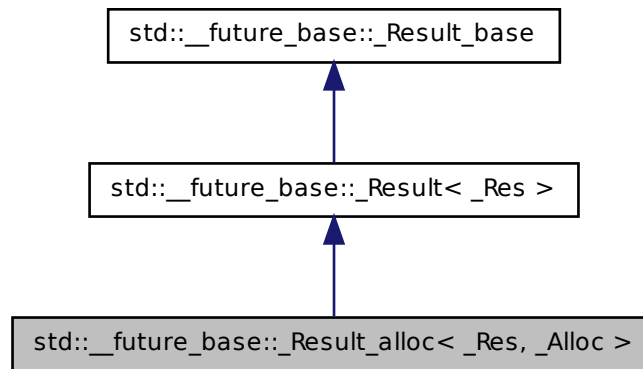
The documentation for this struct was generated from the following file:

- [future](#)

### 5.264 `std::__future_base::_Result_alloc< _Res, _Alloc >` Struct Template Reference

`Result_alloc`.

Inheritance diagram for `std::__future_base::__Result_alloc<_Res, _Alloc>`:



### Public Types

- `typedef _Alloc::template rebind< \_Result\_alloc >::other __allocator_type`

### Public Member Functions

- `_Result_alloc (const _Alloc &__a)`
- `void _M_set (_Res &&__res)`
- `void _M_set (const _Res &__res)`
- `_Res & _M_value ()`

### Public Attributes

- `exception_ptr _M_error`

#### 5.264.1 Detailed Description

`template<typename _Res, typename _Alloc> struct std::__future_base::__Result_alloc<_Res, _Alloc>`

`Result_alloc.`

Definition at line 238 of file `future`.

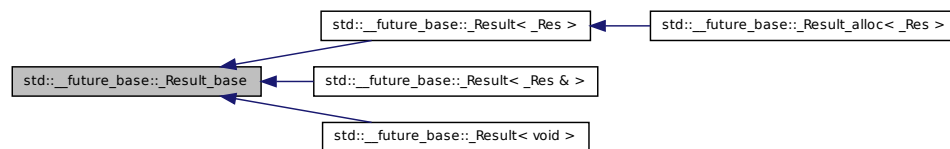
The documentation for this struct was generated from the following file:

- [future](#)

## 5.265 `std::__future_base::_Result_base` Struct Reference

Base class for results.

Inheritance diagram for `std::__future_base::_Result_base`:



### Public Member Functions

- `_Result_base` (const [\\_Result\\_base](#) &)
- virtual void `_M_destroy` ()=0
- [\\_Result\\_base](#) & `operator=` (const [\\_Result\\_base](#) &)

### Public Attributes

- `exception_ptr` `_M_error`

#### 5.265.1 Detailed Description

Base class for results.

Definition at line 159 of file `future`.

The documentation for this struct was generated from the following file:

- [future](#)

## 5.266 `std::__future_base::_State` Class Reference

Shared state between a promise and one or more associated futures.

Inherited by `std::__future_base::_Async_state< _Res >`, `std::__future_base::_Deferred_state< _Res >`, and `std::__future_base::_Task_state< _Res(_Args...)>`.

### Public Member Functions

- `_State` (const `_State` &)
- `void _M_break_promise` (`_Ptr_type` \_\_res)
- `void _M_set_result` (function< `_Ptr_type`()> \_\_res, bool \_\_ignore\_failure=false)
- `void _M_set_retrieved_flag` ()
- `_State` & `operator=` (const `_State` &)
- `_Result_base` & `wait` ()
- `template<typename _Rep, typename _Period >`  
`bool wait_for` (const `chrono::duration`< `_Rep`, `_Period` > &\_\_rel)
- `template<typename _Clock, typename _Duration >`  
`bool wait_until` (const `chrono::time_point`< `_Clock`, `_Duration` > &\_\_abs)

### Static Public Member Functions

- `template<typename _Res, typename _Arg >`  
`static _Setter< _Res, _Arg && > __setter` (`promise`< `_Res` > \*\_\_prom, `_Arg` &&\_\_arg)
- `template<typename _Res >`  
`static _Setter< _Res, __exception_ptr_tag > __setter` (`exception_ptr` &\_\_ex, `promise`< `_Res` > \*\_\_prom)
- `static _Setter< void, void > __setter` (`promise`< void > \*\_\_prom)
- `template<typename _Tp >`  
`static bool _S_check` (const `shared_ptr`< `_Tp` > &\_\_p)

#### 5.266.1 Detailed Description

Shared state between a promise and one or more associated futures.

Definition at line 277 of file `future`.

The documentation for this class was generated from the following file:

- `future`

## 5.267 `std::__has_iterator_category_helper< _Tp >` Class Template Reference 1492

### 5.267 `std::__has_iterator_category_helper< _Tp >` Class Template Reference

Traits class for iterators.

Inherits `std::__sfn_`.

#### Static Public Attributes

- static const bool `value`

#### Private Types

- typedef char `__one`

#### 5.267.1 Detailed Description

`template<typename _Tp> class std::__has_iterator_category_helper< _Tp >`

Traits class for iterators. This class does nothing but define nested typedefs. The general version simply *forwards* the nested typedefs from the Iterator argument. Specialized versions for pointers and pointers-to-const provide tighter, more correct semantics.

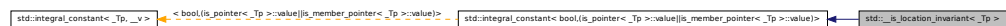
Definition at line 143 of file `stl_iterator_base_types.h`.

The documentation for this class was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

### 5.268 `std::__is_location_invariant< _Tp >` Struct Template Reference

Inheritance diagram for `std::__is_location_invariant< _Tp >`:



#### Public Types

- typedef [integral\\_constant](#)< bool, `__v` > `type`
- typedef bool `value_type`

## 5.269 `std::__is_member_pointer_helper<_Tp>` Struct Template Reference 1493

### Public Member Functions

- `constexpr operator value_type()`

### Static Public Attributes

- `static constexpr bool value`

#### 5.268.1 Detailed Description

`template<typename _Tp> struct std::__is_location_invariant<_Tp>`

Trait identifying "location-invariant" types, meaning that the address of the object (or any of its members) will not escape. Also implies a trivial copy constructor and assignment operator.

Definition at line 1484 of file `functional`.

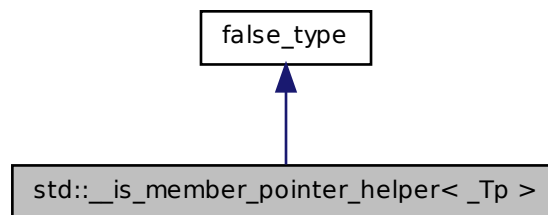
The documentation for this struct was generated from the following file:

- [functional](#)

## 5.269 `std::__is_member_pointer_helper<_Tp>` Struct Template Reference

`is_member_pointer`

Inheritance diagram for `std::__is_member_pointer_helper<_Tp>`:





### Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr \_Tp **value**

#### 5.269.1 Detailed Description

**template<typename \_Tp> struct std::\_\_is\_member\_pointer\_helper< \_Tp >**

**is\_member\_pointer**

Definition at line 312 of file `type_traits`.

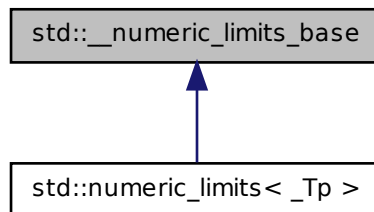
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.270 `std::__numeric_limits_base` Struct Reference

Part of [std::numeric\\_limits](#).

Inheritance diagram for `std::__numeric_limits_base`:



### Static Public Attributes

- static constexpr int [digits](#)
- static constexpr int [digits10](#)
- static constexpr float\_denorm\_style [has\\_denorm](#)
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool [has\\_infinity](#)
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool [is\\_bounded](#)
- static constexpr bool [is\\_exact](#)
- static constexpr bool [is\\_iec559](#)
- static constexpr bool [is\\_integer](#)
- static constexpr bool [is\\_modulo](#)
- static constexpr bool [is\\_signed](#)
- static constexpr bool [is\\_specialized](#)
- static constexpr int [max\\_digits10](#)
- static constexpr int [max\\_exponent](#)
- static constexpr int [max\\_exponent10](#)
- static constexpr int [min\\_exponent](#)
- static constexpr int [min\\_exponent10](#)
- static constexpr int [radix](#)
- static constexpr float\_round\_style [round\\_style](#)
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool [traps](#)

#### 5.270.1 Detailed Description

Part of [std::numeric\\_limits](#). The `static const` members are usable as integral constant expressions.

#### Note

This is a separate class for purposes of efficiency; you should only access these members as part of an instantiation of the [std::numeric\\_limits](#) class.

Definition at line 192 of file `limits`.

## 5.270.2 Member Data Documentation

### 5.270.2.1 `constexpr int std::__numeric_limits_base::digits` `[static]`

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

Definition at line 201 of file `limits`.

### 5.270.2.2 `constexpr int std::__numeric_limits_base::digits10` `[static]`

The number of base 10 digits that can be represented without change.

Definition at line 204 of file `limits`.

### 5.270.2.3 `constexpr float_denorm_style std::__numeric_limits_base::has_denorm` `[static]`

See [std::float\\_denorm\\_style](#) for more information.

Definition at line 258 of file `limits`.

### 5.270.2.4 `constexpr bool std::__numeric_limits_base::has_denorm_loss` `[static]`

*True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result. [18.2.1.2]/42*

Definition at line 262 of file `limits`.

### 5.270.2.5 `constexpr bool std::__numeric_limits_base::has_infinity` `[static]`

True if the type has a representation for positive infinity.

Definition at line 247 of file `limits`.

### 5.270.2.6 `constexpr bool std::__numeric_limits_base::has_quiet_NaN` `[static]`

True if the type has a representation for a quiet (non-signaling) *Not a Number*.

Definition at line 251 of file `limits`.

**5.270.2.7 constexpr bool std::\_\_numeric\_limits\_base::has\_signaling\_NaN [static]**

True if the type has a representation for a signaling *Not a Number*.

Definition at line 255 of file limits.

**5.270.2.8 constexpr bool std::\_\_numeric\_limits\_base::is\_bounded [static]**

*True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types. [18.2.1.2]/54*

Definition at line 271 of file limits.

**5.270.2.9 constexpr bool std::\_\_numeric\_limits\_base::is\_exact [static]**

True if the type uses an exact representation. *All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer. [18.2.1.2]/15*

Definition at line 223 of file limits.

**5.270.2.10 constexpr bool std::\_\_numeric\_limits\_base::is\_iec559 [static]**

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

Definition at line 266 of file limits.

**5.270.2.11 constexpr bool std::\_\_numeric\_limits\_base::is\_integer [static]**

True if the type is integer. Is this supposed to be *if the type is integral*?

Definition at line 217 of file limits.

**5.270.2.12 constexpr bool std::\_\_numeric\_limits\_base::is\_modulo [static]**

True if the type is *modulo*, that is, if it is possible to add two positive numbers and have a result that wraps around to a third number that is less. Typically false for floating types, true for unsigned integers, and true for signed integers.

Definition at line 277 of file limits.

**5.270.2.13 `constexpr bool std::__numeric_limits_base::is_signed [static]`**

True if the type is signed.

Definition at line 213 of file limits.

**5.270.2.14 `constexpr bool std::__numeric_limits_base::is_specialized [static]`**

This will be true for all fundamental types (which have specializations), and false for everything else.

Definition at line 196 of file limits.

**5.270.2.15 `constexpr int std::__numeric_limits_base::max_digits10 [static]`**

The number of base 10 digits required to ensure that values which differ are always differentiated.

Definition at line 209 of file limits.

**5.270.2.16 `constexpr int std::__numeric_limits_base::max_exponent [static]`**

The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

Definition at line 240 of file limits.

**5.270.2.17 `constexpr int std::__numeric_limits_base::max_exponent10 [static]`**

The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

Definition at line 244 of file limits.

**5.270.2.18 `constexpr int std::__numeric_limits_base::min_exponent [static]`**

The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

Definition at line 231 of file limits.

**5.270.2.19 `constexpr int std::__numeric_limits_base::min_exponent10`  
`[static]`**

The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

Definition at line 235 of file limits.

**5.270.2.20 `constexpr int std::__numeric_limits_base::radix` `[static]`**

For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

Definition at line 227 of file limits.

**5.270.2.21 `constexpr float_round_style std::__numeric_limits_base::round_style` `[static]`**

See [std::float\\_round\\_style](#) for more information. This is only meaningful for floating types; integer types will all be `round_toward_zero`.

Definition at line 288 of file limits.

**5.270.2.22 `constexpr bool std::__numeric_limits_base::tinyness_before`  
`[static]`**

True if tininess is detected before rounding. (see IEC 559)

Definition at line 283 of file limits.

**5.270.2.23 `constexpr bool std::__numeric_limits_base::traps` `[static]`**

True if trapping is implemented for this type.

Definition at line 280 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

**5.271 `std::__parallel::_CRandNumber< _MustBeInt >` Struct Template Reference**

Functor wrapper for `std::rand()`.

**Public Member Functions**

- `int operator() (int __limit)`

**5.271.1 Detailed Description**

`template<typename _MustBeInt = int> struct std::__parallel::CRandNumber<_MustBeInt>`

Functor wrapper for `std::rand()`.

Definition at line 1656 of file `algo.h`.

The documentation for this struct was generated from the following file:

- [algo.h](#)

**5.272 `std::__profile::bitset<_Nb>` Class Template Reference**

Class [std::bitset](#) wrapper with performance instrumentation.

Inherits `bitset<_Nb>`.

**Public Member Functions**

- `constexpr bitset (unsigned long long __val)`
- `template<class _CharT, class _Traits, class _Alloc> bitset (const std::basic_string<_CharT, _Traits, _Alloc> &__str, typename std::basic_string<_CharT, _Traits, _Alloc>::size_type __pos, typename std::basic_string<_CharT, _Traits, _Alloc>::size_type __n, _CharT __zero, _CharT __one=_CharT('1'))`
- `bitset (const _Base &__x)`
- `template<typename _CharT, typename _Traits, typename _Alloc> bitset (const std::basic_string<_CharT, _Traits, _Alloc> &__str, typename std::basic_string<_CharT, _Traits, _Alloc>::size_type __pos=0, typename std::basic_string<_CharT, _Traits, _Alloc>::size_type __n=(std::basic_string<_CharT, _Traits, _Alloc>::npos))`
- `template<typename _CharT> bitset (const _CharT *__str, typename std::basic_string<_CharT>::size_type __n=std::basic_string<_CharT>::npos, _CharT __zero=_CharT('0'), _CharT __one=_CharT('1'))`
- `_Base & _M_base ()`
- `const _Base & _M_base () const`
- `bitset<_Nb> & flip ()`

- [bitset](#)<\_Nb> & **flip** (size\_t \_\_pos)
- bool **operator!=** (const [bitset](#)<\_Nb> &\_\_rhs) const
- [bitset](#)<\_Nb> & **operator&=** (const [bitset](#)<\_Nb> &\_\_rhs)
- [bitset](#)<\_Nb> **operator<<** (size\_t \_\_pos) const
- [bitset](#)<\_Nb> & **operator<<=** (size\_t \_\_pos)
- bool **operator==** (const [bitset](#)<\_Nb> &\_\_rhs) const
- [bitset](#)<\_Nb> **operator>>** (size\_t \_\_pos) const
- [bitset](#)<\_Nb> & **operator>>=** (size\_t \_\_pos)
- reference **operator[ ]** (size\_t \_\_pos)
- bool **operator[ ]** (size\_t \_\_pos) const
- [bitset](#)<\_Nb> & **operator^=** (const [bitset](#)<\_Nb> &\_\_rhs)
- [bitset](#)<\_Nb> & **operator|=** (const [bitset](#)<\_Nb> &\_\_rhs)
- [bitset](#)<\_Nb> **operator~** () const
- [bitset](#)<\_Nb> & **reset** ()
- [bitset](#)<\_Nb> & **reset** (size\_t \_\_pos)
- [bitset](#)<\_Nb> & **set** (size\_t \_\_pos, bool \_\_val=true)
- [bitset](#)<\_Nb> & **set** ()
- [std::basic\\_string](#)< char, [std::char\\_traits](#)< char >, [std::allocator](#)< char > > **to\_string** (char \_\_zero, char \_\_one= '1') const
- template<typename \_CharT, typename \_Traits >  
[std::basic\\_string](#)< \_CharT, \_Traits, [std::allocator](#)< \_CharT > > **to\_string** () const
- template<typename \_CharT, typename \_Traits, typename \_Alloc >  
[std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > **to\_string** () const
- template<class \_CharT, class \_Traits >  
[std::basic\\_string](#)< \_CharT, \_Traits, [std::allocator](#)< \_CharT > > **to\_string** (\_CharT \_\_zero, \_CharT \_\_one=\_CharT('1')) const
- [std::basic\\_string](#)< char, [std::char\\_traits](#)< char >, [std::allocator](#)< char > > **to\_string** () const
- template<typename \_CharT >  
[std::basic\\_string](#)< \_CharT, [std::char\\_traits](#)< \_CharT >, [std::allocator](#)< \_CharT > > **to\_string** () const
- template<class \_CharT >  
[std::basic\\_string](#)< \_CharT, [std::char\\_traits](#)< \_CharT >, [std::allocator](#)< \_CharT > > **to\_string** (\_CharT \_\_zero, \_CharT \_\_one=\_CharT('1')) const
- template<class \_CharT, class \_Traits, class \_Alloc >  
[std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > **to\_string** (\_CharT \_\_zero, \_CharT \_\_one=\_CharT('1')) const

### 5.272.1 Detailed Description

template<size\_t \_Nb> class std::\_\_profile::bitset<\_Nb>

Class [std::bitset](#) wrapper with performance instrumentation.



## 5.273 `std::__profile::deque< _Tp, _Allocator >` Class Template Reference 1502

---

Definition at line 40 of file `profile/bitset`.

The documentation for this class was generated from the following file:

- [profile/bitset](#)

### 5.273 `std::__profile::deque< _Tp, _Allocator >` Class Template Reference

Class [std::deque](#) wrapper with performance instrumentation.

Inherits `deque< _Tp, _Allocator >`.

#### Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `_Base::const_iterator` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `_Base::const_reverse_iterator` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `_Base::iterator` **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `_Base::reverse_iterator` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Tp` **value\_type**

#### Public Member Functions

- **deque** (`const _Allocator &__a=_Allocator()`)
- **deque** (`size_type __n`)
- `template<class _InputIterator >`  
**deque** (`_InputIterator __first, _InputIterator __last, const _Allocator &__a=_Allocator()`)
- **deque** ([initializer\\_list](#)< `value_type` > `__l`, `const allocator_type &__a=allocator_type()`)
- **deque** (`const deque &__x`)
- **deque** (`size_type __n, const _Tp &__value, const _Allocator &__a=_Allocator()`)
- **deque** (`const \_Base &__x`)
- **deque** ([deque](#) &&`__x`)
- [\\_Base](#) & **\_M\_base** ()

- `const _Base & _M_base () const`
- `template<class _InputIterator >`  
`void assign (_InputIterator __first, _InputIterator __last)`
- `void assign (size_type __n, const _Tp &__t)`
- `void assign (initializer_list< value_type > __l)`
- `reference back ()`
- `const_reference back () const`
- `iterator begin ()`
- `const_iterator begin () const`
- `const_iterator cbegin () const`
- `const_iterator cend () const`
- `void clear ()`
- `const_reverse_iterator crbegin () const`
- `const_reverse_iterator crend () const`
- `template<typename... _Args>`  
`iterator emplace (iterator __position, _Args &&...__args)`
- `template<typename... _Args>`  
`void emplace_back (_Args &&...__args)`
- `template<typename... _Args>`  
`void emplace_front (_Args &&...__args)`
- `iterator end ()`
- `const_iterator end () const`
- `iterator erase (iterator __first, iterator __last)`
- `iterator erase (iterator __position)`
- `reference front ()`
- `const_reference front () const`
- `iterator insert (iterator __position, _Tp &&__x)`
- `void insert (iterator __p, initializer_list< value_type > __l)`
- `void insert (iterator __position, size_type __n, const _Tp &__x)`
- `iterator insert (iterator __position, const _Tp &__x)`
- `template<class _InputIterator >`  
`void insert (iterator __position, _InputIterator __first, _InputIterator __last)`
- `deque & operator= (const deque &__x)`
- `deque & operator= (deque &&__x)`
- `deque & operator= (initializer_list< value_type > __l)`
- `const_reference operator[] (size_type __n) const`
- `reference operator[] (size_type __n)`
- `void pop_back ()`
- `void pop_front ()`
- `void push_back (const _Tp &__x)`
- `void push_back (_Tp &&__x)`
- `void push_front (_Tp &&__x)`
- `void push_front (const _Tp &__x)`

## 5.274 `std::__profile::forward_list<_Tp, _Alloc>` Class Template Reference 504

- [reverse\\_iterator](#) `rbegin()`
- `const_reverse_iterator` `rbegin()` `const`
- `const_reverse_iterator` `rend()` `const`
- [reverse\\_iterator](#) `rend()`
- `void` `resize` (`size_type` `__sz`, `const` `_Tp` `&__c`)
- `void` `resize` (`size_type` `__sz`)
- `void` `swap` ([deque](#) `&__x`)

### 5.273.1 Detailed Description

`template<typename _Tp, typename _Allocator = std::allocator<_Tp>> class std::__profile::deque<_Tp, _Allocator>`

Class [std::deque](#) wrapper with performance instrumentation.

Definition at line 40 of file `profile/deque`.

The documentation for this class was generated from the following file:

- [profile/deque](#)

## 5.274 `std::__profile::forward_list<_Tp, _Alloc>` Class Template Reference

Class [std::forward\\_list](#) wrapper with performance instrumentation.

Inherits `forward_list<_Tp, _Alloc>`.

### Public Types

- `typedef` `_Base::size_type` `size_type`

### Public Member Functions

- `forward_list` (`const` `_Alloc` `&__al`=`_Alloc()`)
- `forward_list` (`const` [forward\\_list](#) `&__list`, `const` `_Alloc` `&__al`)
- `forward_list` (`size_type` `__n`)
- `forward_list` ([forward\\_list](#) `&&__list`)
- `forward_list` ([std::initializer\\_list](#)<`_Tp`> `__il`, `const` `_Alloc` `&__al`=`_Alloc()`)
- `forward_list` (`size_type` `__n`, `const` `_Tp` `&__value`, `const` `_Alloc` `&__al`=`_Alloc()`)
- `forward_list` ([forward\\_list](#) `&&__list`, `const` `_Alloc` `&__al`)

- `template<typename _InputIterator>`  
`forward_list` (`_InputIterator __first`, `_InputIterator __last`, `const _Alloc &__al=_Alloc()`)
- `forward_list` (`const forward_list &__list`)
- `const _Base & _M_base () const`
- `_Base & _M_base ()`
- `forward_list & operator= (std::initializer_list<_Tp> __il)`
- `forward_list & operator= (const forward_list &__list)`
- `forward_list & operator= (forward_list &&__list)`

#### 5.274.1 Detailed Description

`template<typename _Tp, typename _Alloc = std::allocator<_Tp>> class std::_profile::forward_list<_Tp, _Alloc>`

Class `std::forward_list` wrapper with performance instrumentation.

Definition at line 44 of file `profile/forward_list`.

The documentation for this class was generated from the following file:

- `profile/forward_list`

### 5.275 `std::__profile::list<_Tp, _Allocator>` Class Template Reference

List wrapper with performance instrumentation.

Inherits `list<_Tp, _Allocator>`.

#### Public Types

- `typedef _Allocator allocator_type`
- `typedef __iterator_tracker< typename _Base::const_iterator, list > const_iterator`
- `typedef _Base::const_pointer const_pointer`
- `typedef _Base::const_reference const_reference`
- `typedef std::reverse_iterator< const_iterator > const_reverse_iterator`
- `typedef _Base::difference_type difference_type`
- `typedef __iterator_tracker< typename _Base::iterator, list > iterator`
- `typedef _Base::pointer pointer`
- `typedef _Base::reference reference`
- `typedef std::reverse_iterator< iterator > reverse_iterator`
- `typedef _Base::size_type size_type`
- `typedef _Tp value_type`

**Public Member Functions**

- **list** (const \_Allocator &\_\_a=\_Allocator())
- **list** (size\_type \_\_n)
- template<class \_InputIterator >  
  **list** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Allocator &\_\_a=\_Allocator())
- **list** (initializer\_list< value\_type > \_\_l, const allocator\_type &\_\_a=allocator\_type())
- **list** (const list &\_\_x)
- **list** (size\_type \_\_n, const \_Tp &\_\_value, const \_Allocator &\_\_a=\_Allocator())
- **list** (const \_Base &\_\_x)
- **list** (list &&\_\_x)
- const \_Base & \_M\_base () const
- \_Base & \_M\_base ()
- void \_M\_profile\_find () const
- void \_M\_profile\_iterate (int \_\_rewind=0) const
- void assign (initializer\_list< value\_type > \_\_l)
- template<class \_InputIterator >  
  void assign (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void assign (size\_type \_\_n, const \_Tp &\_\_t)
- const\_reference back () const
- reference back ()
- const\_iterator begin () const
- iterator begin ()
- const\_iterator cbegin () const
- const\_iterator cend () const
- void clear ()
- const\_reverse\_iterator crbegin () const
- const\_reverse\_iterator crend () const
- template<typename... \_Args>  
  iterator emplace (iterator \_\_position, \_Args &&...\_\_args)
- iterator end ()
- const\_iterator end () const
- iterator erase (iterator \_\_position)
- iterator erase (iterator \_\_position, iterator \_\_last)
- const\_reference front () const
- reference front ()
- iterator insert (iterator \_\_position, const \_Tp &\_\_x)
- iterator insert (iterator \_\_position, \_Tp &&\_\_x)
- void insert (iterator \_\_position, size\_type \_\_n, const \_Tp &\_\_x)
- void insert (iterator \_\_position, initializer\_list< value\_type > \_\_l)

- `template<class _InputIterator >`  
`void insert (iterator __position, _InputIterator __first, _InputIterator __last)`
- `template<class _Compare >`  
`void merge (list &&__x, _Compare __comp)`
- `template<typename _Compare >`  
`void merge (list &__x, _Compare __comp)`
- `void merge (list &&__x)`
- `void merge (list &__x)`
- `list & operator= (const list &__x)`
- `list & operator= (list &&__x)`
- `list & operator= (initializer_list< value_type > __l)`
- `void pop_back ()`
- `void pop_front ()`
- `void push_front (const value_type &__x)`
- `const_reverse_iterator rbegin () const`
- `reverse_iterator rbegin ()`
- `void remove (const _Tp &__value)`
- `template<class _Predicate >`  
`void remove_if (_Predicate __pred)`
- `const_reverse_iterator rend () const`
- `reverse_iterator rend ()`
- `void resize (size_type __sz, const _Tp &__c)`
- `void resize (size_type __sz)`
- `template<typename _StrictWeakOrdering >`  
`void sort (_StrictWeakOrdering __pred)`
- `void sort ()`
- `void splice (iterator __position, list &&__x)`
- `void splice (iterator __position, list &&__x, iterator __first, iterator __last)`
- `void splice (iterator __position, list &__x, iterator __i)`
- `void splice (iterator __position, list &&__x, iterator __i)`
- `void splice (iterator __position, list &__x, iterator __first, iterator __last)`
- `void splice (iterator __position, list &__x)`
- `void swap (list &__x)`
- `template<class _BinaryPredicate >`  
`void unique (_BinaryPredicate __binary_pred)`
- `void unique ()`

### 5.275.1 Detailed Description

`template<typename _Tp, typename _Allocator = std::allocator<_Tp>> class std::__profile::list<_Tp, _Allocator>`

List wrapper with performance instrumentation.

Definition at line 42 of file `profile/list`.

The documentation for this class was generated from the following file:

- [profile/list](#)

## 5.276 `std::__profile::map< _Key, _Tp, _Compare, _Allocator >` Class Template Reference

Class [std::map](#) wrapper with performance instrumentation.

Inherits `map< _Key, _Tp, _Compare, _Allocator >`.

### Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `_Base::const_iterator` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `_Base::iterator` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Tp` **mapped\_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `std::pair< const _Key, _Tp >` **value\_type**

### Public Member Functions

- **map** (`const _Compare &__comp=_Compare()`, `const _Allocator &__a=_Allocator()`)
- `template<typename _InputIterator >`  
**map** (`_InputIterator __first, _InputIterator __last, const _Compare &__comp=_Compare()`, `const _Allocator &__a=_Allocator()`)
- **map** (`const _Base &__x`)
- **map** (`map &&__x`)
- **map** (`const map &__x`)
- **map** (`initializer_list< value_type > __l, const _Compare &__c=_Compare()`, `const allocator_type &__a=allocator_type()`)

- `_Base` & `_M_base` ()
- `const _Base` & `_M_base` () const
- `mapped_type` & `at` (const `key_type` &\_\_k)
- `const mapped_type` & `at` (const `key_type` &\_\_k) const
- `iterator begin` ()
- `const_iterator begin` () const
- `const_iterator cbegin` () const
- `const_iterator cend` () const
- `void clear` ()
- `size_type count` (const `key_type` &\_\_x) const
- `const_reverse_iterator crbegin` () const
- `const_reverse_iterator crend` () const
- `iterator end` ()
- `const_iterator end` () const
- `std::pair< iterator, iterator > equal_range` (const `key_type` &\_\_x)
- `std::pair< const_iterator, const_iterator > equal_range` (const `key_type` &\_\_x) const
- `iterator erase` (const `iterator` \_\_first, const `iterator` \_\_last)
- `iterator erase` (const `iterator` \_\_position)
- `size_type erase` (const `key_type` &\_\_x)
- `iterator find` (const `key_type` &\_\_x)
- `const_iterator find` (const `key_type` &\_\_x) const
- `template<typename _InputIterator > void insert` (\_InputIterator \_\_first, \_InputIterator \_\_last)
- `template<typename _Pair, typename = typename std::enable_if<std::is_convertible<_Pair, value_type>::value>::type> iterator insert` (const `iterator` \_\_position, \_Pair &&\_\_x)
- `std::pair< iterator, bool > insert` (const `value_type` &\_\_x)
- `template<typename _Pair, typename = typename std::enable_if<std::is_convertible<_Pair, value_type>::value>::type> std::pair< iterator, bool > insert` (\_Pair &&\_\_x)
- `void insert` (std::initializer\_list< `value_type` > \_\_list)
- `iterator insert` (const `iterator` \_\_position, const `value_type` &\_\_x)
- `const_iterator lower_bound` (const `key_type` &\_\_x) const
- `iterator lower_bound` (const `key_type` &\_\_x)
- `map` & `operator=` (map &&\_\_x)
- `map` & `operator=` (const map &\_\_x)
- `map` & `operator=` (initializer\_list< `value_type` > \_\_l)
- `mapped_type` & `operator[]` (const `key_type` &\_\_k)
- `mapped_type` & `operator[]` (`key_type` &&\_\_k)
- `reverse_iterator rbegin` ()
- `const_reverse_iterator rbegin` () const
- `const_reverse_iterator rend` () const



- [reverse\\_iterator](#) **rend** ()
- void **swap** ([map](#) &\_\_x)
- [iterator](#) **upper\_bound** (const key\_type &\_\_x)
- const\_iterator **upper\_bound** (const key\_type &\_\_x) const

#### 5.276.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<-
_Key>, typename _Allocator = std::allocator<std::pair<const _Key, _Tp> >>
class std::__profile::map< _Key, _Tp, _Compare, _Allocator >
```

Class [std::map](#) wrapper with performance instrumentation.

Definition at line 47 of file profile/map.h.

The documentation for this class was generated from the following file:

- [profile/map.h](#)

#### 5.277 **std::\_\_profile::multimap< \_Key, \_Tp, \_Compare, \_Allocator > Class Template Reference**

Class [std::multimap](#) wrapper with performance instrumentation.

Inherits `multimap< _Key, _Tp, _Compare, _Allocator >`.

#### Public Types

- typedef \_Allocator **allocator\_type**
- typedef \_Base::const\_iterator **const\_iterator**
- typedef \_Base::const\_pointer **const\_pointer**
- typedef \_Base::const\_reference **const\_reference**
- typedef \_Base::const\_reverse\_iterator **const\_reverse\_iterator**
- typedef \_Base::difference\_type **difference\_type**
- typedef \_Base::iterator **iterator**
- typedef \_Compare **key\_compare**
- typedef \_Key **key\_type**
- typedef \_Tp **mapped\_type**
- typedef \_Base::pointer **pointer**
- typedef \_Base::reference **reference**
- typedef \_Base::reverse\_iterator **reverse\_iterator**
- typedef \_Base::size\_type **size\_type**
- typedef [std::pair](#)< const \_Key, \_Tp > **value\_type**

## Public Member Functions

- **multimap** (const \_Compare &\_\_comp=\_Compare(), const \_Allocator &\_\_a=\_Allocator())
- template<typename \_InputIterator >  
**multimap** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Compare &\_\_comp=\_Compare(), const \_Allocator &\_\_a=\_Allocator())
- **multimap** (const [\\_Base](#) &\_\_x)
- **multimap** ([multimap](#) &&\_\_x)
- **multimap** (const [multimap](#) &\_\_x)
- **multimap** ([initializer\\_list](#)< [value\\_type](#) > \_\_l, const \_Compare &\_\_c=\_Compare(), const allocator\_type &\_\_a=allocator\_type())
- [\\_Base](#) & [\\_M\\_base](#) ()
- const [\\_Base](#) & [\\_M\\_base](#) () const
- [iterator](#) **begin** ()
- const\_iterator **begin** () const
- const\_iterator **cbegin** () const
- const\_iterator **end** () const
- void **clear** ()
- const\_reverse\_iterator **crbegin** () const
- const\_reverse\_iterator **crend** () const
- [iterator](#) **end** ()
- const\_iterator **end** () const
- [std::pair](#)< [iterator](#), [iterator](#) > **equal\_range** (const key\_type &\_\_x)
- [std::pair](#)< const\_iterator, const\_iterator > **equal\_range** (const key\_type &\_\_x) const
- [iterator](#) **erase** (const\_iterator \_\_first, const\_iterator \_\_last)
- [iterator](#) **erase** (const\_iterator \_\_position)
- size\_type **erase** (const key\_type &\_\_x)
- [iterator](#) **find** (const key\_type &\_\_x)
- const\_iterator **find** (const key\_type &\_\_x) const
- [iterator](#) **insert** (const [value\\_type](#) &\_\_x)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_convertible<\_Pair, value\_type>::value>::type>  
[iterator](#) **insert** (\_Pair &&\_\_x)
- void **insert** ([std::initializer\\_list](#)< [value\\_type](#) > \_\_list)
- [iterator](#) **insert** (const\_iterator \_\_position, const [value\\_type](#) &\_\_x)
- template<typename \_InputIterator >  
 void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_convertible<\_Pair, value\_type>::value>::type>  
[iterator](#) **insert** (const\_iterator \_\_position, \_Pair &&\_\_x)
- [iterator](#) **lower\_bound** (const key\_type &\_\_x)

- `const_iterator` **lower\_bound** (`const key_type &__x`) `const`
- `multimap` & **operator=** (`multimap &&__x`)
- `multimap` & **operator=** (`const multimap &__x`)
- `multimap` & **operator=** (`initializer_list< value_type > __l`)
- `reverse_iterator` **rbegin** ()
- `const_reverse_iterator` **rbegin** () `const`
- `const_reverse_iterator` **rend** () `const`
- `reverse_iterator` **rend** ()
- `void` **swap** (`multimap &__x`)
- `const_iterator` **upper\_bound** (`const key_type &__x`) `const`
- `iterator` **upper\_bound** (`const key_type &__x`)

#### 5.277.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>,
typename _Allocator = std::allocator<std::pair<const _Key, _Tp> >>
class std::__profile::multimap< _Key, _Tp, _Compare, _Allocator >
```

Class `std::multimap` wrapper with performance instrumentation.

Definition at line 41 of file `profile/multimap.h`.

The documentation for this class was generated from the following file:

- [profile/multimap.h](#)

## 5.278 `std::__profile::multiset< _Key, _Compare, _Allocator >` Class Template Reference

Class `std::multiset` wrapper with performance instrumentation.

Inherits `multiset< _Key, _Compare, _Allocator >`.

#### Public Types

- `typedef _Allocator` **allocator\_type**
- `typedef _Base::const_iterator` **const\_iterator**
- `typedef _Base::const_pointer` **const\_pointer**
- `typedef _Base::const_reference` **const\_reference**
- `typedef _Base::const_reverse_iterator` **const\_reverse\_iterator**
- `typedef _Base::difference_type` **difference\_type**
- `typedef _Base::iterator` **iterator**
- `typedef _Compare` **key\_compare**

- typedef `_Key` **key\_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `_Base::reverse_iterator` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Compare` **value\_compare**
- typedef `_Key` **value\_type**

### Public Member Functions

- **multiset** (const `_Compare` &\_\_comp=\_Compare(), const `_Allocator` &\_\_a=\_Allocator())
- template<typename `_InputIterator` >  
**multiset** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Compare` &\_\_comp=\_Compare(), const `_Allocator` &\_\_a=\_Allocator())
- **multiset** (const `_Base` &\_\_x)
- **multiset** (`multiset` &&\_\_x)
- **multiset** (const `multiset` &\_\_x)
- **multiset** (`initializer_list`< `value_type` > \_\_l, const `_Compare` &\_\_comp=\_Compare(), const `allocator_type` &\_\_a=allocator\_type())
- `_Base` & **\_M\_base** ()
- const `_Base` & **\_M\_base** () const
- `iterator` **begin** ()
- const `iterator` **begin** () const
- const `iterator` **cbegin** () const
- const `iterator` **cend** () const
- void **clear** ()
- const `reverse_iterator` **crbegin** () const
- const `reverse_iterator` **crend** () const
- `iterator` **end** ()
- const `iterator` **end** () const
- `std::pair`< `iterator`, `iterator` > **equal\_range** (const `key_type` &\_\_x)
- `std::pair`< const `iterator`, const `iterator` > **equal\_range** (const `key_type` &\_\_x) const
- `iterator` **erase** (const `iterator` \_\_first, const `iterator` \_\_last)
- `iterator` **erase** (const `iterator` \_\_position)
- `size_type` **erase** (const `key_type` &\_\_x)
- `iterator` **find** (const `key_type` &\_\_x)
- const `iterator` **find** (const `key_type` &\_\_x) const
- `iterator` **insert** (const `value_type` &\_\_x)
- `iterator` **insert** (`value_type` &&\_\_x)
- `iterator` **insert** (const `iterator` \_\_position, const `value_type` &\_\_x)
- `iterator` **insert** (const `iterator` \_\_position, `value_type` &&\_\_x)

- void **insert** ([initializer\\_list](#)< value\_type > \_\_l)
- template<typename \_InputIterator >  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- [iterator](#) **lower\_bound** (const key\_type &\_\_x)
- const\_iterator **lower\_bound** (const key\_type &\_\_x) const
- [multiset](#) & **operator=** ([multiset](#) &&\_\_x)
- [multiset](#) & **operator=** (const [multiset](#) &\_\_x)
- [multiset](#) & **operator=** ([initializer\\_list](#)< value\_type > \_\_l)
- [reverse\\_iterator](#) **rbegin** ()
- const\_reverse\_iterator **rbegin** () const
- const\_reverse\_iterator **rend** () const
- [reverse\\_iterator](#) **rend** ()
- void **swap** ([multiset](#) &\_\_x)
- const\_iterator **upper\_bound** (const key\_type &\_\_x) const
- [iterator](#) **upper\_bound** (const key\_type &\_\_x)

#### 5.278.1 Detailed Description

template<typename \_Key, typename \_Compare = std::less<\_Key>, typename \_Allocator = std::allocator<\_Key>> class std::\_\_profile::multiset< \_Key, \_Compare, \_Allocator >

Class [std::multiset](#) wrapper with performance instrumentation.

Definition at line 41 of file profile/multiset.h.

The documentation for this class was generated from the following file:

- [profile/multiset.h](#)

#### 5.279 `std::__profile::set< _Key, _Compare, _Allocator >` Class Template Reference

Class [std::set](#) wrapper with performance instrumentation.

Inherits `set< _Key, _Compare, _Allocator >`.

#### Public Types

- typedef \_Allocator **allocator\_type**
- typedef \_Base::const\_iterator **const\_iterator**
- typedef \_Base::const\_pointer **const\_pointer**
- typedef \_Base::const\_reference **const\_reference**

- `typedef _Base::const_reverse_iterator` **const\_reverse\_iterator**
- `typedef _Base::difference_type` **difference\_type**
- `typedef _Base::iterator` **iterator**
- `typedef _Compare` **key\_compare**
- `typedef _Key` **key\_type**
- `typedef _Base::pointer` **pointer**
- `typedef _Base::reference` **reference**
- `typedef _Base::reverse_iterator` **reverse\_iterator**
- `typedef _Base::size_type` **size\_type**
- `typedef _Compare` **value\_compare**
- `typedef _Key` **value\_type**

### Public Member Functions

- **set** (const `_Compare` &\_\_comp=\_Compare(), const `_Allocator` &\_\_a=\_Allocator())
- `template<typename _InputIterator>`  
**set** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Compare` &\_\_comp=\_Compare(), const `_Allocator` &\_\_a=\_Allocator())
- **set** (const `_Base` &\_\_x)
- **set** (`set` &&\_\_x)
- **set** (const `set` &\_\_x)
- **set** (`initializer_list`< `value_type` > \_\_l, const `_Compare` &\_\_comp=\_Compare(), const `allocator_type` &\_\_a=allocator\_type())
- `_Base` & **\_M\_base** ()
- const `_Base` & **\_M\_base** () const
- `iterator` **begin** ()
- const `iterator` **begin** () const
- const `iterator` **cbegin** () const
- const `iterator` **end** () const
- void **clear** ()
- const `reverse_iterator` **crbegin** () const
- const `reverse_iterator` **crend** () const
- `iterator` **end** ()
- const `iterator` **end** () const
- `std::pair`< `iterator`, `iterator` > **equal\_range** (const `key_type` &\_\_x)
- `std::pair`< const `iterator`, const `iterator` > **equal\_range** (const `key_type` &\_\_x) const
- `iterator` **erase** (const `iterator` \_\_first, const `iterator` \_\_last)
- `iterator` **erase** (const `iterator` \_\_position)
- `size_type` **erase** (const `key_type` &\_\_x)
- `iterator` **find** (const `key_type` &\_\_x)
- const `iterator` **find** (const `key_type` &\_\_x) const

- [std::pair](#)< [iterator](#), bool > **insert** (const value\_type &\_\_x)
- [std::pair](#)< [iterator](#), bool > **insert** (value\_type &&\_\_x)
- [iterator](#) **insert** (const\_iterator \_\_position, const value\_type &\_\_x)
- [iterator](#) **insert** (const\_iterator \_\_position, value\_type &&\_\_x)
- void **insert** ([initializer\\_list](#)< value\_type > \_\_l)
- template<typename \_InputIterator >  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- [iterator](#) **lower\_bound** (const key\_type &\_\_x)
- const\_iterator **lower\_bound** (const key\_type &\_\_x) const
- [set](#) & **operator=** ([set](#) &&\_\_x)
- [set](#) & **operator=** (const [set](#) &\_\_x)
- [set](#) & **operator=** ([initializer\\_list](#)< value\_type > \_\_l)
- [reverse\\_iterator](#) **rbegin** ()
- const\_reverse\_iterator **rbegin** () const
- const\_reverse\_iterator **rend** () const
- [reverse\\_iterator](#) **rend** ()
- void **swap** ([set](#) &\_\_x)
- const\_iterator **upper\_bound** (const key\_type &\_\_x) const
- [iterator](#) **upper\_bound** (const key\_type &\_\_x)

#### 5.279.1 Detailed Description

template<typename \_Key, typename \_Compare = std::less<\_Key>, typename \_Allocator = std::allocator<\_Key>> class std::\_\_profile::set< \_Key, \_Compare, \_Allocator >

Class [std::set](#) wrapper with performance instrumentation.

Definition at line 41 of file profile/set.h.

The documentation for this class was generated from the following file:

- [profile/set.h](#)

#### 5.280 **std::\_\_profile::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >** **Class Template Reference**

Class [std::unordered\\_map](#) wrapper with performance instrumentation.

Inherits [unordered\\_map](#)< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >.

## Public Types

- `typedef _Base::allocator_type allocator_type`
- `typedef _Base::const_iterator const_iterator`
- `typedef _Base::const_reference const_reference`
- `typedef _Base::difference_type difference_type`
- `typedef _Base::hasher hasher`
- `typedef _Base::iterator iterator`
- `typedef _Base::key_equal key_equal`
- `typedef _Base::key_type key_type`
- `typedef _Base::mapped_type mapped_type`
- `typedef _Base::reference reference`
- `typedef _Base::size_type size_type`
- `typedef _Base::value_type value_type`

## Public Member Functions

- **unordered\_map** (size\_type \_\_n=10, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- `template<typename _InputIterator >`  
**unordered\_map** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_map** ([unordered\\_map](#) &&\_\_x)
- **unordered\_map** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_map** (const [\\_Base](#) &\_\_x)
- [\\_Base](#) & [\\_M\\_base](#) ()
- const [\\_Base](#) & [\\_M\\_base](#) () const
- void **clear** ()
- [iterator](#) **insert** (const\_iterator \_\_iter, const value\_type &\_\_v)
- `template<typename _Pair, typename = typename std::enable_if<std::is_convertible<_Pair, value_type>::value>::type>`  
[std::pair](#)< [iterator](#), bool > **insert** (\_Pair &&\_\_obj)
- `template<typename _Pair, typename = typename std::enable_if<std::is_convertible<_Pair, value_type>::value>::type>`  
[iterator](#) **insert** (const\_iterator \_\_iter, \_Pair &&\_\_v)
- `template<typename _InputIter >`  
void **insert** (\_InputIter \_\_first, \_InputIter \_\_last)
- void **insert** (const value\_type \* \_\_first, const value\_type \* \_\_last)
- void **insert** ([std::initializer\\_list](#)< value\_type > \_\_l)
- [std::pair](#)< [iterator](#), bool > **insert** (const value\_type &\_\_obj)



## 5.281 `std::__profile::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>` Class Template Reference 1518

---

- [unordered\\_map](#) & **operator=** ([initializer\\_list](#)< value\_type > \_\_l)
- [unordered\\_map](#) & **operator=** ([unordered\\_map](#) &&\_\_x)
- [unordered\\_map](#) & **operator=** (const [unordered\\_map](#) &\_\_x)
- mapped\_type & **operator[]** (\_Key &&\_\_k)
- mapped\_type & **operator[]** (const \_Key &\_\_k)
- void **rehash** (size\_type \_\_n)
- void **swap** ([unordered\\_map](#) &\_\_x)

### 5.280.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>,
typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<_Key>> class std::__profile::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>
```

Class [std::unordered\\_map](#) wrapper with performance instrumentation.

Definition at line 56 of file `profile/unordered_map`.

The documentation for this class was generated from the following file:

- [profile/unordered\\_map](#)

## 5.281 `std::__profile::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>` Class Template Reference

Class [std::unordered\\_multimap](#) wrapper with performance instrumentation.

Inherits `unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>`.

### Public Types

- typedef `_Base::allocator_type` **allocator\_type**
- typedef `_Base::const_iterator` **const\_iterator**
- typedef `_Base::const_reference` **const\_reference**
- typedef `_Base::difference_type` **difference\_type**
- typedef `_Base::hasher` **hasher**
- typedef `_Base::iterator` **iterator**
- typedef `_Base::key_equal` **key\_equal**
- typedef `_Base::key_type` **key\_type**
- typedef `_Base::reference` **reference**
- typedef `_Base::size_type` **size\_type**
- typedef `_Base::value_type` **value\_type**

## Public Member Functions

- **unordered\_multimap** (size\_type \_\_n=10, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator >  
**unordered\_multimap** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_multimap** (unordered\_multimap &&\_\_x)
- **unordered\_multimap** (initializer\_list< value\_type > \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_multimap** (const \_Base &\_\_x)
- \_Base & \_M\_base ()
- const \_Base & \_M\_base () const
- void **clear** ()
- iterator **insert** (const\_iterator \_\_iter, const value\_type &\_\_v)
- void **insert** (const value\_type \*\_\_first, const value\_type \*\_\_last)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_convertible<\_Pair, value\_type>::value>::type>  
 iterator **insert** (\_Pair &&\_\_obj)
- template<typename \_InputIter >  
 void **insert** (\_InputIter \_\_first, \_InputIter \_\_last)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_convertible<\_Pair, value\_type>::value>::type>  
 iterator **insert** (const\_iterator \_\_iter, \_Pair &&\_\_v)
- void **insert** (std::initializer\_list< value\_type > \_\_l)
- iterator **insert** (const value\_type &\_\_obj)
- unordered\_multimap & **operator=** (const unordered\_multimap &\_\_x)
- unordered\_multimap & **operator=** (initializer\_list< value\_type > \_\_l)
- unordered\_multimap & **operator=** (unordered\_multimap &&\_\_x)
- void **rehash** (size\_type \_\_n)
- void **swap** (unordered\_multimap &\_\_x)

### 5.281.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>,
typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<_Key>>
class std::__profile::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>
```

Class `std::unordered_multimap` wrapper with performance instrumentation.

Definition at line 330 of file `profile/unordered_map`.

The documentation for this class was generated from the following file:

- [profile/unordered\\_map](#)

## 5.282 `std::__profile::unordered_multiset<_Value, _Hash, _Pred, _Alloc>` Class Template Reference

Unordered\_multiset wrapper with performance instrumentation.

Inherits `_GLIBCXX_STD_BASE`.

### Public Types

- `typedef _Base::allocator_type allocator_type`
- `typedef _Base::const_iterator const_iterator`
- `typedef _Base::const_reference const_reference`
- `typedef _Base::difference_type difference_type`
- `typedef _Base::hasher hasher`
- `typedef _Base::iterator iterator`
- `typedef _Base::key_equal key_equal`
- `typedef _Base::key_type key_type`
- `typedef _Base::reference reference`
- `typedef _Base::size_type size_type`
- `typedef _Base::value_type value_type`

### Public Member Functions

- **unordered\_multiset** (`size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- `template<typename _InputIterator>`  
**unordered\_multiset** (`_InputIterator __f`, `_InputIterator __l`, `size_type __n=0`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- **unordered\_multiset** ([unordered\\_multiset](#) &&\_\_x)
- **unordered\_multiset** ([initializer\\_list](#)< `value_type` > \_\_l, `size_type __n=0`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- **unordered\_multiset** (`const _Base &__x`)
- `void clear ()`
- `template<typename _InputIter>`  
`void insert (_InputIter __first, _InputIter __last)`
- [iterator](#) **insert** (`const_iterator __iter`, `value_type &&__v`)
- `void insert (const value_type *__first, const value_type *__last)`
- [iterator](#) **insert** (`value_type &&__obj`)

- void **insert** ([std::initializer\\_list](#)< value\_type > \_\_l)
- [iterator](#) **insert** (const value\_type &\_\_obj)
- [iterator](#) **insert** (const\_iterator \_\_iter, const value\_type &\_\_v)
- [unordered\\_multiset](#) & **operator=** (const [unordered\\_multiset](#) &\_\_x)
- [unordered\\_multiset](#) & **operator=** ([unordered\\_multiset](#) &&\_\_x)
- [unordered\\_multiset](#) & **operator=** ([initializer\\_list](#)< value\_type > \_\_l)
- void **rehash** (size\_type \_\_n)
- void **swap** ([unordered\\_multiset](#) &\_\_x)

#### 5.282.1 Detailed Description

```
template<typename _Value, typename _Hash = std::hash<_Value>, typename
_Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>>
class std::__profile::unordered_multiset<_Value, _Hash, _Pred, _Alloc>
```

Unordered\_multiset wrapper with performance instrumentation.

Definition at line 302 of file `profile/unordered_set`.

The documentation for this class was generated from the following file:

- [profile/unordered\\_set](#)

### 5.283 `std::__profile::unordered_set<_Key, _Hash, _Pred, _Alloc>` Class Template Reference

Unordered\_set wrapper with performance instrumentation.

Inherits `_GLIBCXX_STD_BASE`.

#### Public Types

- typedef `_Base::allocator_type` **allocator\_type**
- typedef `_Base::const_iterator` **const\_iterator**
- typedef `_Base::const_reference` **const\_reference**
- typedef `_Base::difference_type` **difference\_type**
- typedef `_Base::hasher` **hasher**
- typedef `_Base::iterator` **iterator**
- typedef `_Base::key_equal` **key\_equal**
- typedef `_Base::key_type` **key\_type**
- typedef `_Base::reference` **reference**
- typedef `_Base::size_type` **size\_type**
- typedef `_Base::value_type` **value\_type**

## Public Member Functions

- **unordered\_set** (size\_type \_\_n=10, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator >  
**unordered\_set** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_set** (unordered\_set &&\_\_x)
- **unordered\_set** (initializer\_list< value\_type > \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_set** (const \_Base &\_\_x)
- void **clear** ()
- template<typename \_InputIter >  
void **insert** (\_InputIter \_\_first, \_InputIter \_\_last)
- **iterator insert** (const\_iterator \_\_iter, value\_type &&\_\_v)
- void **insert** (const value\_type \*\_\_first, const value\_type \*\_\_last)
- **std::pair< iterator, bool > insert** (value\_type &&\_\_obj)
- void **insert** (std::initializer\_list< value\_type > \_\_l)
- **std::pair< iterator, bool > insert** (const value\_type &\_\_obj)
- **iterator insert** (const\_iterator \_\_iter, const value\_type &\_\_v)
- **unordered\_set & operator=** (const unordered\_set &\_\_x)
- **unordered\_set & operator=** (unordered\_set &&\_\_x)
- **unordered\_set & operator=** (initializer\_list< value\_type > \_\_l)
- void **rehash** (size\_type \_\_n)
- void **swap** (unordered\_set &\_\_x)

### 5.283.1 Detailed Description

```
template<typename _Key, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<_Key>> class
std::__profile::unordered_set<_Key,_Hash,_Pred,_Alloc>
```

Unordered\_set wrapper with performance instrumentation.

Definition at line 56 of file profile/unordered\_set.

The documentation for this class was generated from the following file:

- [profile/unordered\\_set](#)

## 5.284 std::\_Base\_bitset< \_Nw > Struct Template Reference

Inheritance diagram for std::\_Base\_bitset< \_Nw >:



### Public Types

- typedef unsigned long **\_WordT**

### Public Member Functions

- constexpr **\_Base\_bitset** (unsigned long long \_\_val)
- size\_t **\_M\_are\_all\_aux** () const
- void **\_M\_do\_and** (const [\\_Base\\_bitset](#)< \_Nw > &\_\_x)
- size\_t **\_M\_do\_count** () const
- size\_t **\_M\_do\_find\_first** (size\_t \_\_not\_found) const
- size\_t **\_M\_do\_find\_next** (size\_t \_\_prev, size\_t \_\_not\_found) const
- void **\_M\_do\_flip** ()
- void **\_M\_do\_left\_shift** (size\_t \_\_shift)
- void **\_M\_do\_or** (const [\\_Base\\_bitset](#)< \_Nw > &\_\_x)
- void **\_M\_do\_reset** ()
- void **\_M\_do\_right\_shift** (size\_t \_\_shift)
- void **\_M\_do\_set** ()
- unsigned long long **\_M\_do\_to\_ullong** () const
- unsigned long **\_M\_do\_to\_ulong** () const
- void **\_M\_do\_xor** (const [\\_Base\\_bitset](#)< \_Nw > &\_\_x)
- const \_WordT \* **\_M\_getdata** () const
- \_WordT **\_M\_getword** (size\_t \_\_pos) const
- \_WordT & **\_M\_getword** (size\_t \_\_pos)
- constexpr \_WordT **\_M\_hiword** () const
- \_WordT & **\_M\_hiword** ()
- bool **\_M\_is\_any** () const
- bool **\_M\_is\_equal** (const [\\_Base\\_bitset](#)< \_Nw > &\_\_x) const

### Static Public Member Functions

- static constexpr \_WordT **\_S\_maskbit** (size\_t \_\_pos)
- static constexpr size\_t **\_S\_whichbit** (size\_t \_\_pos)
- static constexpr size\_t **\_S\_whichbyte** (size\_t \_\_pos)
- static constexpr size\_t **\_S\_whichword** (size\_t \_\_pos)

**Public Attributes**

- `_WordT \_M\_w [\_Nw]`

**5.284.1 Detailed Description**

`template<size_t \_Nw> struct std::_Base_bitset< \_Nw >`

Base class, general case. It is a class invariant that `_Nw` will be nonnegative.

See documentation for `bitset`.

Definition at line 71 of file `bitset`.

**5.284.2 Member Data Documentation**

**5.284.2.1** `template<size_t \_Nw> \_WordT std::_Base_bitset< \_Nw >::\_M\_w[\_Nw]`

0 is the least significant word.

Definition at line 76 of file `bitset`.

The documentation for this struct was generated from the following file:

- [bitset](#)

**5.285 `std::_Base_bitset< 0 >` Struct Template Reference****Public Types**

- `typedef unsigned long \_WordT`

**Public Member Functions**

- `constexpr \_Base\_bitset (unsigned long long)`
- `size_t \_M\_are\_all\_aux () const`
- `void \_M\_do\_and (const \_Base\_bitset< 0 > &)`
- `size_t \_M\_do\_count () const`
- `size_t \_M\_do\_find\_first (size_t) const`
- `size_t \_M\_do\_find\_next (size_t, size_t) const`
- `void \_M\_do\_flip ()`
- `void \_M\_do\_left\_shift (size_t)`

- `void _M_do_or (const \_Base\_bitset< 0 > &)`
- `void _M_do_reset ()`
- `void _M_do_right_shift (size_t)`
- `void _M_do_set ()`
- `unsigned long long _M_do_to_ullong () const`
- `unsigned long _M_do_to_ulong () const`
- `void _M_do_xor (const \_Base\_bitset< 0 > &)`
- `_WordT _M_getword (size_t __pos) const`
- `_WordT & _M_getword (size_t)`
- `constexpr _WordT _M_hiword () const`
- `bool _M_is_any () const`
- `bool _M_is_equal (const \_Base\_bitset< 0 > &) const`

#### Static Public Member Functions

- `static constexpr _WordT _S_maskbit (size_t __pos)`
- `static constexpr size_t _S_whichbit (size_t __pos)`
- `static constexpr size_t _S_whichbyte (size_t __pos)`
- `static constexpr size_t _S_whichword (size_t __pos)`

#### 5.285.1 Detailed Description

**template<> struct `std::_Base_bitset< 0 >`**

Base class, specialization for no storage (zero-length bitset).

See documentation for bitset.

Definition at line 512 of file bitset.

The documentation for this struct was generated from the following file:

- [bitset](#)

## 5.286 `std::_Base_bitset< 1 >` Struct Template Reference

### Public Types

- `typedef unsigned long _WordT`



**Public Member Functions**

- constexpr **\_Base\_bitset** (unsigned long long \_\_val)
- size\_t **\_M\_are\_all\_aux** () const
- void **\_M\_do\_and** (const [\\_Base\\_bitset](#)< 1 > &\_\_x)
- size\_t **\_M\_do\_count** () const
- size\_t **\_M\_do\_find\_first** (size\_t \_\_not\_found) const
- size\_t **\_M\_do\_find\_next** (size\_t \_\_prev, size\_t \_\_not\_found) const
- void **\_M\_do\_flip** ()
- void **\_M\_do\_left\_shift** (size\_t \_\_shift)
- void **\_M\_do\_or** (const [\\_Base\\_bitset](#)< 1 > &\_\_x)
- void **\_M\_do\_reset** ()
- void **\_M\_do\_right\_shift** (size\_t \_\_shift)
- void **\_M\_do\_set** ()
- unsigned long long **\_M\_do\_to\_ullong** () const
- unsigned long **\_M\_do\_to\_ulong** () const
- void **\_M\_do\_xor** (const [\\_Base\\_bitset](#)< 1 > &\_\_x)
- const \_WordT \* **\_M\_getdata** () const
- \_WordT **\_M\_getword** (size\_t) const
- \_WordT & **\_M\_getword** (size\_t)
- constexpr \_WordT **\_M\_hiword** () const
- \_WordT & **\_M\_hiword** ()
- bool **\_M\_is\_any** () const
- bool **\_M\_is\_equal** (const [\\_Base\\_bitset](#)< 1 > &\_\_x) const

**Static Public Member Functions**

- static constexpr \_WordT **\_S\_maskbit** (size\_t \_\_pos)
- static constexpr size\_t **\_S\_whichbit** (size\_t \_\_pos)
- static constexpr size\_t **\_S\_whichbyte** (size\_t \_\_pos)
- static constexpr size\_t **\_S\_whichword** (size\_t \_\_pos)

**Public Attributes**

- \_WordT **\_M\_w**

### 5.286.1 Detailed Description

**template<> struct std::\_Base\_bitset< 1 >**

Base class, specialization for a single word.

See documentation for `bitset`.

Definition at line 368 of file `bitset`.

The documentation for this struct was generated from the following file:

- [bitset](#)

## 5.287 `std::_Build_index_tuple<_Num>` Struct Template Reference

Builds an `_Index_tuple<0, 1, 2, ..., _Num-1>`.

### Public Types

- typedef `_Build_index_tuple<_Num-1>::__type::__next __type`

### 5.287.1 Detailed Description

**template<std::size\_t \_Num> struct std::\_Build\_index\_tuple<\_Num>**

Builds an `_Index_tuple<0, 1, 2, ..., _Num-1>`.

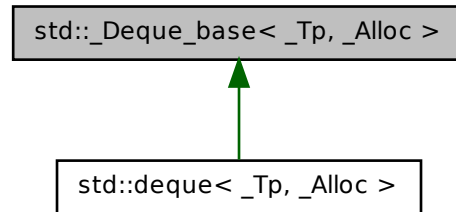
Definition at line 771 of file `tuple`.

The documentation for this struct was generated from the following file:

- [tuple](#)

**5.288 std::\_Deque\_base< \_Tp, \_Alloc > Class Template Reference**

Inheritance diagram for std::\_Deque\_base< \_Tp, \_Alloc >:

**Public Types**

- typedef `_Alloc` **allocator\_type**
- typedef `_Deque_iterator< _Tp, const _Tp &, const _Tp * >` **const\_iterator**
- typedef `_Deque_iterator< _Tp, _Tp &, _Tp * >` **iterator**

**Public Member Functions**

- `_Deque_base` (const allocator\_type &\_\_a, size\_t \_\_num\_elements)
- `_Deque_base` (`_Deque_base` &&\_\_x)
- `_Deque_base` (const allocator\_type &\_\_a)
- `_Deque_base` (size\_t \_\_num\_elements)
- allocator\_type **get\_allocator** () const

**Protected Types**

- enum { `_S_initial_map_size` }
- typedef `_Alloc::template rebind< _Tp * >::other` **\_Map\_alloc\_type**
- typedef `_Alloc::template rebind< _Tp >::other` **\_Tp\_alloc\_type**

**Protected Member Functions**

- `_Tp **` **\_M\_allocate\_map** (size\_t \_\_n)

- `_Tp * _M_allocate_node ()`
- `void _M_create_nodes (_Tp **__nstart, _Tp **__nfinish)`
- `void _M_deallocate_map (_Tp **__p, size_t __n)`
- `void _M_deallocate_node (_Tp *__p)`
- `void _M_destroy_nodes (_Tp **__nstart, _Tp **__nfinish)`
- `_Map_alloc_type _M_get_map_allocator () const`
- `const _Tp_alloc_type & _M_get_Tp_allocator () const`
- `_Tp_alloc_type & _M_get_Tp_allocator ()`
- `void _M_initialize_map (size_t)`

### Protected Attributes

- `_Deque_impl _M_impl`

### 5.288.1 Detailed Description

**template<typename \_Tp, typename \_Alloc> class std::Deque\_base< \_Tp, \_Alloc >**

Deque base class. This class provides the unified face for deque's allocation. This class's constructor and destructor allocate and deallocate (but do not initialize) storage. This makes exception safety easier.

Nothing in this class ever constructs or destroys an actual Tp element. (Deque handles that itself.) Only/All memory management is performed here.

Definition at line 438 of file stl\_deque.h.

### 5.288.2 Member Function Documentation

**5.288.2.1 template<typename \_Tp , typename \_Alloc > void  
std::Deque\_base< \_Tp, \_Alloc >::\_M\_initialize\_map ( size\_t  
\_\_num\_elements ) [protected]**

Layout storage.

#### Parameters

*num\_elements* The count of T's for which to allocate space at first.

#### Returns

Nothing.

## 5.289 `std::_Deque_iterator<_Tp, _Ref, _Ptr>` Struct Template Reference 1530

The initial underlying memory layout is a bit complicated...

Definition at line 574 of file `stl_deque.h`.

References `std::max()`.

Referenced by `std::deque<_Tp, _Alloc>::_M_range_initialize()`.

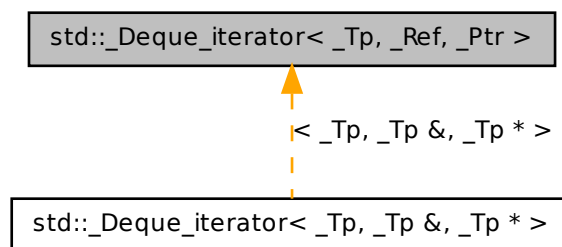
The documentation for this class was generated from the following file:

- [stl\\_deque.h](#)

## 5.289 `std::_Deque_iterator<_Tp, _Ref, _Ptr>` Struct Template Reference

A `deque::iterator`.

Inheritance diagram for `std::_Deque_iterator<_Tp, _Ref, _Ptr>`:



### Public Types

- `typedef _Tp** _Map_pointer`
- `typedef _Deque_iterator _Self`
- `typedef _Deque_iterator<_Tp, const _Tp &, const _Tp*> const_iterator`
- `typedef ptrdiff_t difference_type`
- `typedef _Deque_iterator<_Tp, _Tp &, _Tp*> iterator`
- `typedef std::random_access_iterator_tag iterator_category`
- `typedef _Ptr pointer`
- `typedef _Ref reference`
- `typedef size_t size_type`

- `typedef _Tp value_type`

#### Public Member Functions

- `_Deque_iterator` (`_Tp *__x`, `_Map_pointer __y`)
- `_Deque_iterator` (`const iterator &__x`)
- `void _M_set_node` (`_Map_pointer __new_node`)
- reference `operator*` (`() const`)
- `_Self operator+` (`difference_type __n`) `const`
- `_Self operator++` (`int`)
- `_Self & operator++` (`()`)
- `_Self & operator+=` (`difference_type __n`)
- `_Self operator-` (`difference_type __n`) `const`
- `_Self & operator--` (`()`)
- `_Self operator--` (`int`)
- `_Self & operator-=` (`difference_type __n`)
- pointer `operator->` (`() const`)
- reference `operator[ ]` (`difference_type __n`) `const`

#### Static Public Member Functions

- `static size_t _S_buffer_size` (`()`)

#### Public Attributes

- `_Tp * _M_cur`
- `_Tp * _M_first`
- `_Tp * _M_last`
- `_Map_pointer _M_node`

##### 5.289.1 Detailed Description

`template<typename _Tp, typename _Ref, typename _Ptr> struct std::_Deque_iterator<_Tp,_Ref,_Ptr>`

A `deque::iterator`. Quite a bit of intelligence here. Much of the functionality of `deque` is actually passed off to this class. A `deque` holds two of these internally, marking its valid range. Access to elements is done as offsets of either of those two, relying on operator overloading in this class.

All the functions are op overloads except for `_M_set_node`.

Definition at line 105 of file `stl_deque.h`.

## 5.290 `std::_Derives_from_binary_function< _Tp >` Struct Template Reference 1532

### 5.289.2 Member Function Documentation

**5.289.2.1** `template<typename _Tp, typename _Ref, typename _Ptr>`  
`void std::_Deque_iterator< _Tp, _Ref, _Ptr >::_M_set_node (`  
`_Map_pointer __new_node ) [inline]`

Prepares to traverse `new_node`. Sets everything except `_M_cur`, which should therefore be set by the caller immediately afterwards, based on `_M_first` and `_M_last`.

Definition at line 233 of file `stl_deque.h`.

The documentation for this struct was generated from the following file:

- [stl\\_deque.h](#)

## 5.290 `std::_Derives_from_binary_function< _Tp >` Struct Template Reference

Determines if the type `_Tp` derives from [binary\\_function](#).

Inherits `std::__sfinae_types`.

### Public Types

- `typedef char __one`

### Static Public Attributes

- `static const bool value`

### 5.290.1 Detailed Description

`template<typename _Tp> struct std::_Derives_from_binary_function< _Tp >`

Determines if the type `_Tp` derives from [binary\\_function](#).

Definition at line 201 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

## **5.291** `std::_Derives_from_unary_function< _Tp >` Struct Template Reference

Determines if the type `_Tp` derives from [unary\\_function](#).

Inherits `std::__sfinae_types`.

### **Public Types**

- typedef char `__one`

### **Static Public Attributes**

- static const bool `value`

#### **5.291.1 Detailed Description**

`template<typename _Tp> struct std::_Derives_from_unary_function< _Tp >`

Determines if the type `_Tp` derives from [unary\\_function](#).

Definition at line 185 of file `functional`.

The documentation for this struct was generated from the following file:

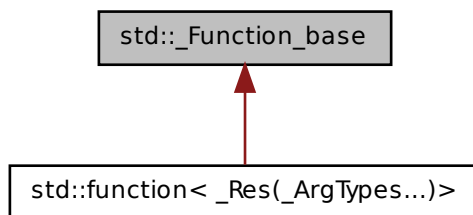
- [functional](#)

## **5.292** `std::_Function_base` Class Reference

Base class of all polymorphic function object wrappers.



Inheritance diagram for `std::_Function_base`:



### Public Types

- `typedef bool(* _Manager_type )(_Any_data &, const _Any_data &, _Manager_operation)`

### Public Member Functions

- `bool _M_empty () const`

### Public Attributes

- `_Any_data _M_functor`
- `_Manager_type _M_manager`

### Static Public Attributes

- `static const std::size_t _M_max_align`
- `static const std::size_t _M_max_size`

#### 5.292.1 Detailed Description

Base class of all polymorphic function object wrappers.

Definition at line 1562 of file `functional`.

The documentation for this class was generated from the following file:

- [functional](#)

**5.293 `std::_Function_to_function_pointer<_Tp, _IsFunctionType > Struct` Template Reference**

Turns a function type into a function pointer type.

**Public Types**

- `typedef _Tp type`

**5.293.1 Detailed Description**

```
template<typename _Tp, bool _IsFunctionType = is_function<_Tp>::value>
struct std::_Function_to_function_pointer<_Tp, _IsFunctionType >
```

Turns a function type into a function pointer type.

Definition at line 217 of file `functional`.

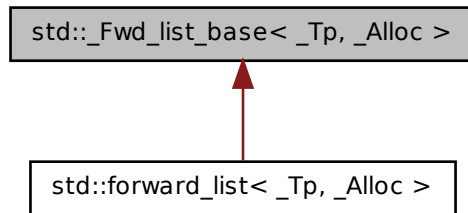
The documentation for this struct was generated from the following file:

- [functional](#)

**5.294 `std::_Fwd_list_base<_Tp, _Alloc > Struct` Template Reference**

Base class for `forward_list`.

Inheritance diagram for `std::_Fwd_list_base<_Tp, _Alloc>`:



### Public Types

- typedef `_Fwd_list_node<_Tp>` **\_Node**
- typedef `_Fwd_list_const_iterator<_Tp>` **const\_iterator**
- typedef `_Fwd_list_iterator<_Tp>` **iterator**

### Public Member Functions

- `_Fwd_list_base` (const `_Alloc` &\_\_a)
- `_Fwd_list_base` (`_Fwd_list_base` &&\_\_lst)
- `_Fwd_list_base` (const `_Fwd_list_base` &\_\_lst, const `_Alloc` &\_\_a)
- `_Fwd_list_base` (`_Fwd_list_base` &&\_\_lst, const `_Alloc` &\_\_a)
- const `_Node_alloc_type` & `_M_get_Node_allocator` () const
- `_Node_alloc_type` & `_M_get_Node_allocator` ()

### Protected Types

- typedef `_Alloc::template rebind<_Fwd_list_node<_Tp>>::other` **\_Node\_alloc\_type**
- typedef `_Alloc::template rebind<_Tp>::other` **\_Tp\_alloc\_type**

### Protected Member Functions

- template<typename... `_Args`>  
`_Node` \* `_M_create_node` (`_Args` &&...\_\_args)

- `_Fwd_list_node_base * _M_erase_after (_Fwd_list_node_base * __pos)`
- `_Fwd_list_node_base * _M_erase_after (_Fwd_list_node_base * __pos, _Fwd_list_node_base * __last)`
- `_Node * _M_get_node ()`
- `template<typename... _Args> _Fwd_list_node_base * _M_insert_after (const_iterator __pos, _Args &&... _args)`
- `void _M_put_node (_Node * __p)`

### Protected Attributes

- `_Fwd_list_impl _M_impl`

#### 5.294.1 Detailed Description

`template<typename _Tp, typename _Alloc> struct std::_Fwd_list_base<_Tp, _Alloc>`

Base class for `forward_list`.

Definition at line 273 of file `forward_list.h`.

The documentation for this struct was generated from the following files:

- `forward_list.h`
- `forward_list.tcc`

### 5.295 `std::_Fwd_list_const_iterator<_Tp>` Struct Template Reference

A `forward_list::const_iterator`.

### Public Types

- `typedef const _Fwd_list_node<_Tp> _Node`
- `typedef _Fwd_list_const_iterator<_Tp> _Self`
- `typedef ptrdiff_t difference_type`
- `typedef _Fwd_list_iterator<_Tp> iterator`
- `typedef std::forward_iterator_tag iterator_category`
- `typedef const _Tp * pointer`
- `typedef const _Tp & reference`
- `typedef _Tp value_type`

**Public Member Functions**

- `_Fwd_list_const_iterator` (const `_Fwd_list_node_base` \*\_\_n)
- `_Fwd_list_const_iterator` (const `iterator` &\_\_iter)
- `_Self` `_M_next` () const
- bool `operator!=` (const `_Self` &\_\_x) const
- reference `operator*` () const
- `_Self` `operator++` (int)
- `_Self` & `operator++` ()
- pointer `operator->` () const
- bool `operator==` (const `_Self` &\_\_x) const

**Public Attributes**

- const `_Fwd_list_node_base` \* `_M_node`

**5.295.1 Detailed Description**

`template<typename _Tp> struct std::_Fwd_list_const_iterator<_Tp>`

A `forward_list::const_iterator`. All the functions are op overloads.

Definition at line 185 of file `forward_list.h`.

The documentation for this struct was generated from the following file:

- [forward\\_list.h](#)

**5.296 `std::_Fwd_list_iterator<_Tp>` Struct Template Reference**

A `forward_list::iterator`.

**Public Types**

- typedef `_Fwd_list_node<_Tp>` `_Node`
- typedef `_Fwd_list_iterator<_Tp>` `_Self`
- typedef `ptrdiff_t` `difference_type`
- typedef `std::forward_iterator_tag` `iterator_category`
- typedef `_Tp` \* `pointer`
- typedef `_Tp` & `reference`
- typedef `_Tp` `value_type`

### Public Member Functions

- `_Fwd_list_iterator` (`_Fwd_list_node_base *__n`)
- `_Self _M_next` () const
- `bool operator!=` (const `_Self` &\_\_x) const
- `reference operator*` () const
- `_Self & operator++` ()
- `_Self operator++` (int)
- `pointer operator->` () const
- `bool operator==` (const `_Self` &\_\_x) const

### Public Attributes

- `_Fwd_list_node_base * _M_node`

#### 5.296.1 Detailed Description

`template<typename _Tp> struct std::_Fwd_list_iterator<_Tp>`

A `forward_list::iterator`. All the functions are op overloads.

Definition at line 117 of file `forward_list.h`.

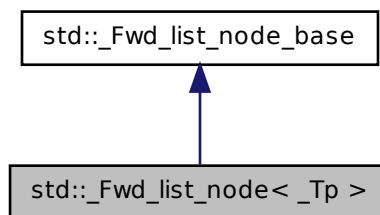
The documentation for this struct was generated from the following file:

- [forward\\_list.h](#)

### 5.297 `std::_Fwd_list_node<_Tp>` Struct Template Reference

A helper node class for `forward_list`. This is just a linked list with a data value in each node. There is a sorting utility method.

Inheritance diagram for std::\_Fwd\_list\_node< \_Tp >:



### Public Member Functions

- `template<typename... _Args>  
_Fwd_list_node (_Args &&...__args)`
- `void _M_reverse_after ()`
- `_Fwd_list_node_base * _M_transfer_after (_Fwd_list_node_base *__begin, _Fwd_list_node_base *__end)`
- `_Fwd_list_node_base * _M_transfer_after (_Fwd_list_node_base *__begin)`

### Public Attributes

- `_Fwd_list_node_base * _M_next`
- `_Tp _M_value`

#### 5.297.1 Detailed Description

**template<typename \_Tp> struct std::\_Fwd\_list\_node< \_Tp >**

A helper node class for forward\_list. This is just a linked list with a data value in each node. There is a sorting utility method.

Definition at line 100 of file forward\_list.h.

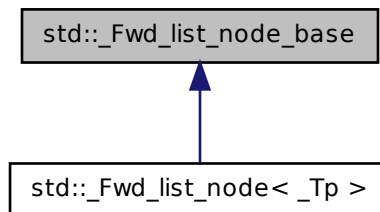
The documentation for this struct was generated from the following file:

- [forward\\_list.h](#)

## 5.298 `std::_Fwd_list_node_base` Struct Reference

A helper basic node class for `forward_list`. This is just a linked list with nothing inside it. There are purely list shuffling utility methods here.

Inheritance diagram for `std::_Fwd_list_node_base`:



### Public Member Functions

- `void _M_reverse_after()`
- `_Fwd_list_node_base * _M_transfer_after(_Fwd_list_node_base * __begin, _Fwd_list_node_base * __end)`
- `_Fwd_list_node_base * _M_transfer_after(_Fwd_list_node_base * __begin)`

### Public Attributes

- `_Fwd_list_node_base * _M_next`

#### 5.298.1 Detailed Description

A helper basic node class for `forward_list`. This is just a linked list with nothing inside it. There are purely list shuffling utility methods here.

Definition at line 47 of file `forward_list.h`.

The documentation for this struct was generated from the following file:

- [forward\\_list.h](#)



## 5.299 `std::_Index_tuple< _Indexes >` Struct Template Reference

### Public Types

- typedef `_Index_tuple< _Indexes..., sizeof...(_Indexes)>` `__next`

### 5.299.1 Detailed Description

`template<int... _Indexes> struct std::_Index_tuple< _Indexes >`

Stores a tuple of indices. Used by `bind()` to extract the elements in a tuple.

Definition at line 764 of file `tuple`.

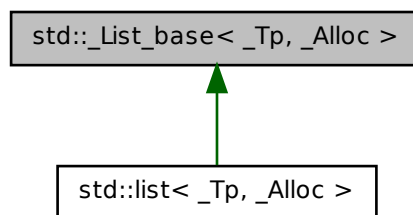
The documentation for this struct was generated from the following file:

- `tuple`

## 5.300 `std::_List_base< _Tp, _Alloc >` Class Template Reference

See `bits/stl_deque.h`'s `_Deque_base` for an explanation.

Inheritance diagram for `std::_List_base< _Tp, _Alloc >`:



### Public Types

- typedef `_Alloc` `allocator_type`

**Public Member Functions**

- `_List_base` (const allocator\_type &\_\_a)
- `_List_base` ([\\_List\\_base](#) &&\_\_x)
- `void _M_clear` ()
- `const _Node_alloc_type & _M_get_Node_allocator` () const
- `_Node_alloc_type & _M_get_Node_allocator` ()
- `_Tp_alloc_type _M_get_Tp_allocator` () const
- `void _M_init` ()
- `allocator_type get_allocator` () const

**Protected Types**

- `typedef _Alloc::template rebind< \_List\_node<_Tp> >::other _Node_alloc_type`
- `typedef _Alloc::template rebind<_Tp>::other _Tp_alloc_type`

**Protected Member Functions**

- `\_List\_node<_Tp> * _M_get_node` ()
- `void _M_put_node` ([\\_List\\_node](#)<\_Tp> \*\_\_p)

**Protected Attributes**

- `_List_impl _M_impl`

**5.300.1 Detailed Description**

`template<typename _Tp, typename _Alloc> class std::_List_base<_Tp, _Alloc>`

See [bits/stl\\_deque.h](#)'s `_Deque_base` for an explanation.

Definition at line 288 of file `stl_list.h`.

The documentation for this class was generated from the following files:

- [stl\\_list.h](#)
- [list.tcc](#)

**5.301 `std::_List_const_iterator<_Tp>` Struct Template Reference**

A `list::const_iterator`.

**Public Types**

- `typedef const _List_node< _Tp > _Node`
- `typedef _List_const_iterator< _Tp > _Self`
- `typedef ptrdiff_t difference_type`
- `typedef _List_iterator< _Tp > iterator`
- `typedef std::bidirectional_iterator_tag iterator_category`
- `typedef const _Tp * pointer`
- `typedef const _Tp & reference`
- `typedef _Tp value_type`

**Public Member Functions**

- `_List_const_iterator` (const `__detail::_List_node_base` \* \_\_x)
- `_List_const_iterator` (const `iterator` & \_\_x)
- `bool operator!=` (const `_Self` & \_\_x) const
- `reference operator*` () const
- `_Self operator++` (int)
- `_Self & operator++` ()
- `_Self & operator--` ()
- `_Self operator--` (int)
- `pointer operator->` () const
- `bool operator==` (const `_Self` & \_\_x) const

**Public Attributes**

- `const __detail::_List_node_base * _M_node`

**5.301.1 Detailed Description**

`template<typename _Tp> struct std::_List_const_iterator< _Tp >`

A `list::const_iterator`. All the functions are op overloads.

Definition at line 199 of file `stl_list.h`.

The documentation for this struct was generated from the following file:

- [stl\\_list.h](#)

**5.302 `std::_List_iterator< _Tp >` Struct Template Reference**

A `list::iterator`.

**Public Types**

- typedef [\\_List\\_node](#)<\_Tp> **\_Node**
- typedef [\\_List\\_iterator](#)<\_Tp> **\_Self**
- typedef ptrdiff\_t **difference\_type**
- typedef [std::bidirectional\\_iterator\\_tag](#) **iterator\_category**
- typedef \_Tp \* **pointer**
- typedef \_Tp & **reference**
- typedef \_Tp **value\_type**

**Public Member Functions**

- [\\_List\\_iterator](#) ([\\_\\_detail::\\_List\\_node\\_base](#) \*\_\_x)
- bool **operator!=** (const [\\_Self](#) &\_\_x) const
- reference **operator\*** () const
- [\\_Self](#) & **operator++** ()
- [\\_Self](#) **operator++** (int)
- [\\_Self](#) & **operator--** ()
- [\\_Self](#) **operator--** (int)
- pointer **operator->** () const
- bool **operator==** (const [\\_Self](#) &\_\_x) const

**Public Attributes**

- [\\_\\_detail::\\_List\\_node\\_base](#) \* **\_M\_node**

**5.302.1 Detailed Description**

**template<typename \_Tp> struct std::\_List\_iterator<\_Tp>**

A list::iterator. All the functions are op overloads.

Definition at line 124 of file `stl_list.h`.

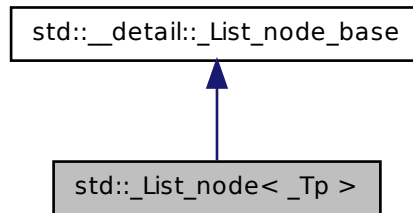
The documentation for this struct was generated from the following file:

- [stl\\_list.h](#)

**5.303 `std::_List_node<_Tp>` Struct Template Reference**

An actual node in the list.

Inheritance diagram for std::\_List\_node< \_Tp >:



### Public Member Functions

- template<typename... \_Args>  
  **\_List\_node** (\_Args &&... \_\_args)
- void **\_M\_hook** (\_List\_node\_base \*const \_\_position) throw ()
- void **\_M\_reverse** () throw ()
- void **\_M\_transfer** (\_List\_node\_base \*const \_\_first, \_List\_node\_base \*const \_\_last) throw ()
- void **\_M\_unhook** () throw ()

### Static Public Member Functions

- static void **swap** (\_List\_node\_base &\_\_x, \_List\_node\_base &\_\_y) throw ()

### Public Attributes

- \_Tp **\_M\_data**
- \_List\_node\_base \* **\_M\_next**
- \_List\_node\_base \* **\_M\_prev**

#### 5.303.1 Detailed Description

**template<typename \_Tp> struct std::\_List\_node< \_Tp >**

An actual node in the list.

Definition at line 105 of file stl\_list.h.

### 5.303.2.1 `template<typename _Tp> _Tp std::List_node< _Tp >::M_data`

Definition at line 108 of file stl\_list.h.

- `stl_list.h`

## > Struct Template Reference

Inheritance diagram for `std::_Maybe_get_result_type<_Has_result_type, _Functor>`:



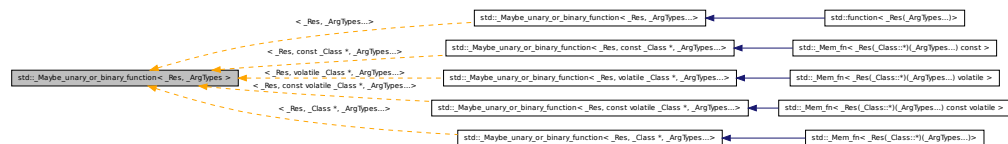
```
template<bool _Has_result_type, typename _Functor> struct std::_Maybe_get_
result_type< _Has_result_type, _Functor >
```

Definition at line 69 of file functional.

- functional

### 5.305 `std::_Maybe_unary_or_binary_function<_Res, _ArgTypes> Struct` Template Reference

Inheritance diagram for `std::_Maybe_unary_or_binary_function<_Res, _ArgTypes>`:



#### 5.305.1 Detailed Description

`template<typename _Res, typename... _ArgTypes> struct std::_Maybe_unary_or_binary_function<_Res, _ArgTypes>`

Derives from `unary_function` or `binary_function`, or perhaps nothing, depending on the number of arguments provided. The primary template is the basis case, which derives nothing.

Definition at line 502 of file `functional`.

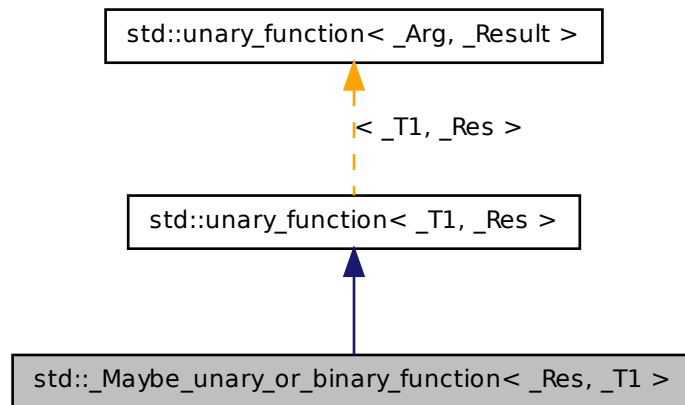
The documentation for this struct was generated from the following file:

- [functional](#)

### 5.306 `std::_Maybe_unary_or_binary_function<_Res, _T1>` Struct Template Reference

Derives from `unary_function`, as appropriate.

Inheritance diagram for `std::_Maybe_unary_or_binary_function<_Res, _T1>`:



## Public Types

- typedef `_T1` [argument\\_type](#)
- typedef `_Res` [result\\_type](#)

### 5.306.1 Detailed Description

`template<typename _Res, typename _T1> struct std::_Maybe_unary_or_binary_function<_Res, _T1>`

Derives from [unary\\_function](#), as appropriate.

Definition at line 506 of file `functional`.

### 5.306.2 Member Typedef Documentation

**5.306.2.1** `typedef _T1 std::unary_function<_T1, _Res>::argument_type`  
`[inherited]`



`argument_type` is the type of the argument

Definition at line 105 of file `std_function.h`.

#### 5.306.2.2 `typedef _Res std::unary_function<_T1, _Res>::result_type` [`inherited`]

`result_type` is the return type

Definition at line 108 of file `std_function.h`.

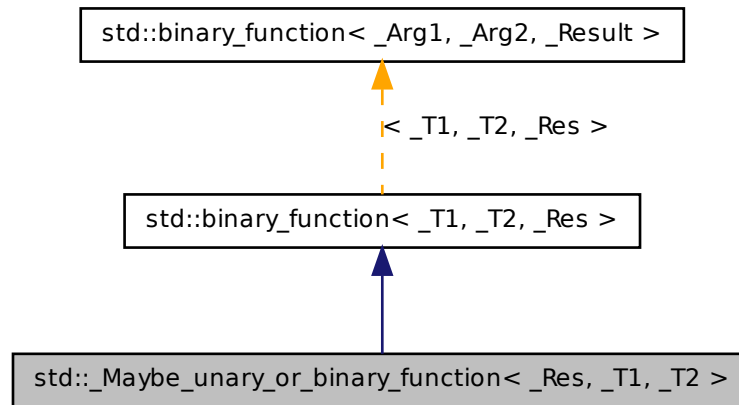
The documentation for this struct was generated from the following file:

- [functional](#)

### 5.307 `std::_Maybe_unary_or_binary_function<_Res, _T1, _T2>` Struct Template Reference

Derives from [binary\\_function](#), as appropriate.

Inheritance diagram for `std::_Maybe_unary_or_binary_function<_Res, _T1, _T2>`:



## Public Types

- `typedef _T1` [first\\_argument\\_type](#)
- `typedef _Res` [result\\_type](#)
- `typedef _T2` [second\\_argument\\_type](#)

### 5.307.1 Detailed Description

`template<typename _Res, typename _T1, typename _T2> struct std::_Maybe_unary_or_binary_function<_Res, _T1, _T2>`

Derives from [binary\\_function](#), as appropriate.

Definition at line 511 of file `functional`.

### 5.307.2 Member Typedef Documentation

**5.307.2.1** `typedef _T1 std::binary_function<_T1, _T2, _Res>::first_argument_type` [\[inherited\]](#)

`first_argument_type` is the type of the first argument

Definition at line 118 of file `stl_function.h`.

**5.307.2.2** `typedef _Res std::binary_function<_T1, _T2, _Res>::result_type` [\[inherited\]](#)

`result_type` is the return type

Definition at line 124 of file `stl_function.h`.

**5.307.2.3** `typedef _T2 std::binary_function<_T1, _T2, _Res>::second_argument_type` [\[inherited\]](#)

`second_argument_type` is the type of the second argument

Definition at line 121 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [functional](#)

## 5.308 `std::_Maybe_wrap_member_pointer<_Tp>` Struct Template Reference 1552

### 5.308 `std::_Maybe_wrap_member_pointer<_Tp>` Struct Template Reference

#### Public Types

- `typedef _Tp type`

#### Static Public Member Functions

- `static const _Tp & __do_wrap (const _Tp &__x)`
- `static _Tp && __do_wrap (_Tp &&__x)`

#### 5.308.1 Detailed Description

`template<typename _Tp> struct std::_Maybe_wrap_member_pointer<_Tp>`

Maps member pointers into instances of `_Mem_fn` but leaves all other function objects untouched. Used by `tr1::bind()`. The primary template handles the non--member-pointer case.

Definition at line 1067 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

### 5.309 `std::_Maybe_wrap_member_pointer<_Tp _Class::*>` Struct Template Reference

#### Public Types

- `typedef _Mem_fn<_Tp _Class::*> type`

#### Static Public Member Functions

- `static type __do_wrap (_Tp _Class::*__pm)`

#### 5.309.1 Detailed Description

`template<typename _Tp, typename _Class> struct std::_Maybe_wrap_member_pointer<_Tp _Class::*>`

Maps member pointers into instances of `_Mem_fn` but leaves all other function objects untouched. Used by `tr1::bind()`. This partial specialization handles the member pointer

case.

Definition at line 1086 of file functional.

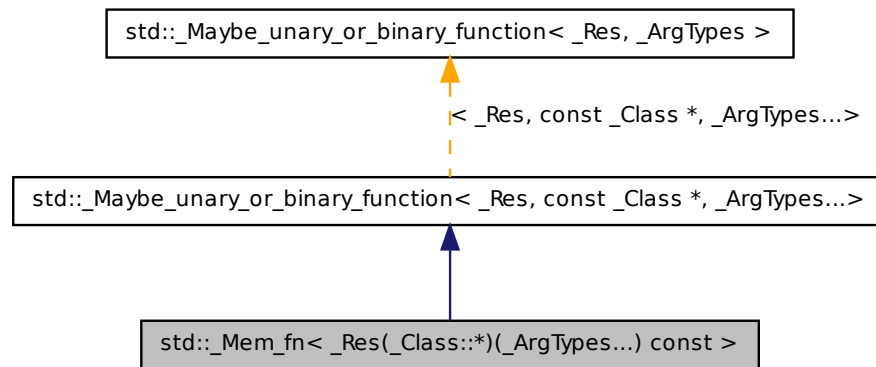
The documentation for this struct was generated from the following file:

- [functional](#)

### 5.310 `std::_Mem_fn<_Res(_Class::*)(_ArgTypes...) const>` Class Template Reference

Implementation of `mem_fn` for const member function pointers.

Inheritance diagram for `std::_Mem_fn<_Res(_Class::*)(_ArgTypes...) const>`:



#### Public Types

- `typedef _Res result_type`

#### Public Member Functions

- `_Mem_fn` (`_Functor __pmf`)
- `template<typename _Tp>`  
`_Res operator() (_Tp &__object, _ArgTypes... __args) const`
- `_Res operator() (const _Class *__object, _ArgTypes... __args) const`
- `_Res operator() (const _Class &__object, _ArgTypes... __args) const`

### 5.310.1 Detailed Description

`template<typename _Res, typename _Class, typename... _ArgTypes> class  
 std::_Mem_fn<_Res(_Class::*)(_ArgTypes...) const >`

Implementation of `mem_fn` for const member function pointers.

Definition at line 562 of file `functional`.

The documentation for this class was generated from the following file:

- [functional](#)

### 5.311 `std::_Mem_fn<_Res(_Class::*)(_ArgTypes...) const volatile > Class` Template Reference

Implementation of `mem_fn` for const volatile member function pointers.

Inheritance diagram for `std::_Mem_fn<_Res(_Class::*)(_ArgTypes...) const volatile >`:



### Public Types

- `typedef _Res result_type`

### Public Member Functions

- `_Mem_fn` (`_Functor __pmf`)
- `template<typename _Tp >  
 _Res operator() (_Tp &__object, _ArgTypes... __args) const`
- `_Res operator() (const volatile _Class *__object, _ArgTypes... __args) const`
- `_Res operator() (const volatile _Class &__object, _ArgTypes... __args) const`

### 5.311.1 Detailed Description

`template<typename _Res, typename _Class, typename... _ArgTypes> class  
 std::_Mem_fn<_Res(_Class::*)(_ArgTypes...) const volatile >`

Implementation of `mem_fn` for const volatile member function pointers.

### 5.312 `std::_Mem_fn<_Res(_Class::*)(_ArgTypes...) volatile >` Class Template Reference

---

1555

Definition at line 655 of file `functional`.

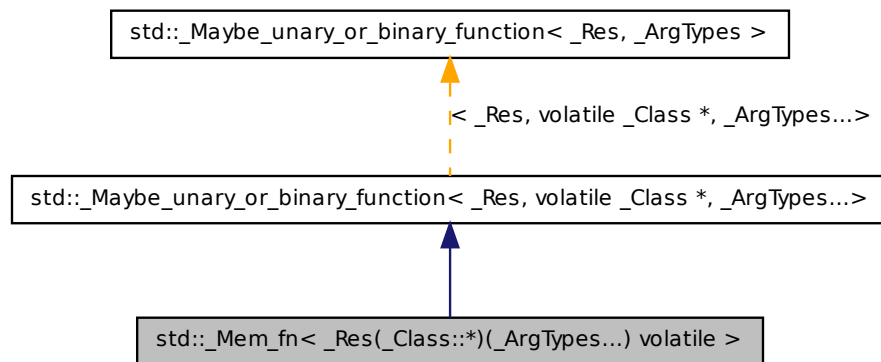
The documentation for this class was generated from the following file:

- [functional](#)

### 5.312 `std::_Mem_fn<_Res(_Class::*)(_ArgTypes...) volatile >` Class Template Reference

Implementation of `mem_fn` for volatile member function pointers.

Inheritance diagram for `std::_Mem_fn<_Res(_Class::*)(_ArgTypes...) volatile >`:



#### Public Types

- `typedef _Res result_type`

#### Public Member Functions

- `_Mem_fn` (`_Functor __pmf`)
- `template<typename _Tp > _Res operator() (_Tp &__object, _ArgTypes... __args) const`
- `_Res operator() (volatile _Class * __object, _ArgTypes... __args) const`
- `_Res operator() (volatile _Class & __object, _ArgTypes... __args) const`

#### 5.312.1 Detailed Description

`template<typename _Res, typename _Class, typename... _ArgTypes> class std::_Mem_fn<_Res(_Class::*)(_ArgTypes...) volatile >`

Implementation of `mem_fn` for volatile member function pointers.

Definition at line 608 of file `functional`.

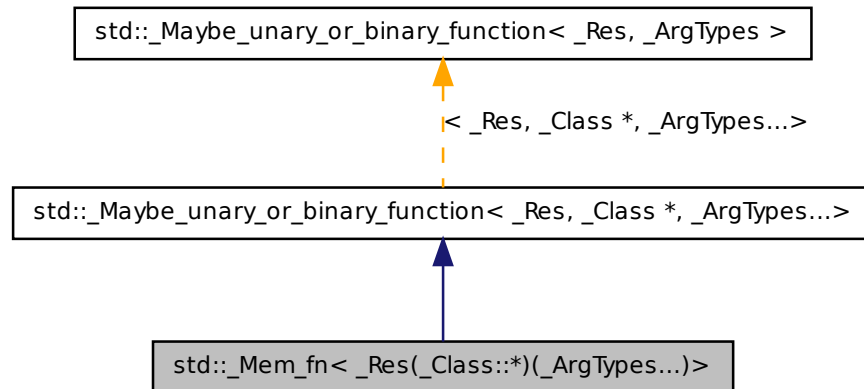
The documentation for this class was generated from the following file:

- [functional](#)

#### 5.313 `std::_Mem_fn<_Res(_Class::*)(_ArgTypes...)>` Class Template Reference

Implementation of `mem_fn` for member function pointers.

Inheritance diagram for `std::_Mem_fn<_Res(_Class::*)(_ArgTypes...)>`:



#### Public Types

- `typedef _Res result_type`

**Public Member Functions**

- `_Mem_fn` (`_Functor __pmf`)
- `template<typename _Tp > _Res operator() (_Tp &__object, _ArgTypes...__args) const`
- `_Res operator() (_Class *__object, _ArgTypes...__args) const`
- `_Res operator() (_Class &__object, _ArgTypes...__args) const`

**5.313.1 Detailed Description**

`template<typename _Res, typename _Class, typename... _ArgTypes> class std::_Mem_fn<_Res(_Class::*)(_ArgTypes...)>`

Implementation of `mem_fn` for member function pointers.

Definition at line 516 of file `functional`.

The documentation for this class was generated from the following file:

- [functional](#)

**5.314 `std::_Mu<_Arg, false, false >` Class Template Reference****Public Member Functions**

- `template<typename _CVarArg, typename _Tuple > _CVarArg && operator() (_CVarArg &&__arg, _Tuple &) const volatile`

**5.314.1 Detailed Description**

`template<typename _Arg> class std::_Mu<_Arg, false, false >`

If the argument is just a value, returns a reference to that value. The cv-qualifiers on the reference are the same as the cv-qualifiers on the `_Mu` object. [TR1 3.6.3/5 bullet 4]

Definition at line 1043 of file `functional`.

The documentation for this class was generated from the following file:

- [functional](#)



## 5.315 `std::_Mu<_Arg, false, true >` Class Template Reference

### Public Member Functions

- `template<typename _Tuple >  
result< _Mu(_Arg, _Tuple)>::type operator() (const volatile _Arg &, _Tuple  
&__tuple) const volatile`

#### 5.315.1 Detailed Description

`template<typename _Arg> class std::_Mu<_Arg, false, true >`

If the argument is a placeholder for the Nth argument, returns a reference to the Nth argument to the bind function object. [TR1 3.6.3/5 bullet 3]

Definition at line 1009 of file functional.

The documentation for this class was generated from the following file:

- [functional](#)

## 5.316 `std::_Mu<_Arg, true, false >` Class Template Reference

### Public Member Functions

- `template<typename _CVArg, typename... _Args>  
auto operator() (_CVArg &__arg, tuple<_Args...> &__tuple) const volatile->  
decltype(__arg(declval<_Args>()...))`

#### 5.316.1 Detailed Description

`template<typename _Arg> class std::_Mu<_Arg, true, false >`

If the argument is a bind expression, we invoke the underlying function object with the same cv-qualifiers as we are given and pass along all of our arguments (unwrapped). [TR1 3.6.3/5 bullet 2]

Definition at line 975 of file functional.

The documentation for this class was generated from the following file:

- [functional](#)

## **5.317 std::\_Mu< reference\_wrapper< \_Tp >, false, false > Class Template Reference**

### **Public Types**

- typedef \_Tp & **result\_type**

### **Public Member Functions**

- template<typename \_CVRef, typename \_Tuple >  
result\_type **operator()** (\_CVRef &\_\_arg, \_Tuple &) const volatile

#### **5.317.1 Detailed Description**

**template<typename \_Tp> class std::\_Mu< reference\_wrapper< \_Tp >, false, false >**

If the argument is reference\_wrapper<\_Tp>, returns the underlying reference. [TR1 3.6.3/5 bullet 1]

Definition at line 954 of file functional.

The documentation for this class was generated from the following file:

- [functional](#)

## **5.318 std::\_Placeholder< \_Num > Struct Template Reference**

The type of placeholder objects defined by libstdc++.

### **5.318.1 Detailed Description**

**template<int \_Num> struct std::\_Placeholder< \_Num >**

The type of placeholder objects defined by libstdc++.

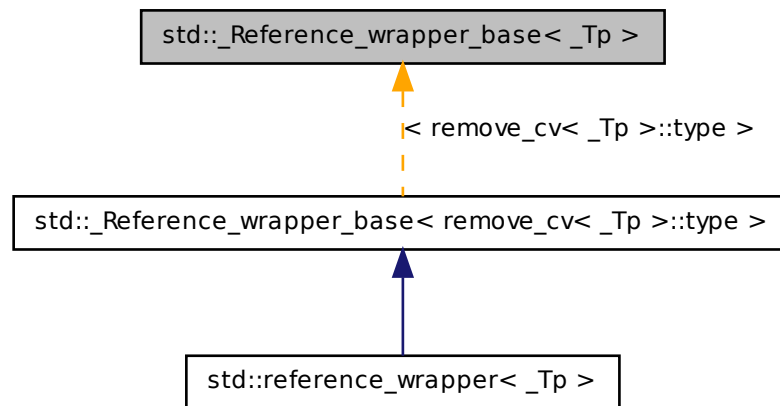
Definition at line 838 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

### 5.319 `std::_Reference_wrapper_base<_Tp>` Struct Template Reference

Inheritance diagram for `std::_Reference_wrapper_base<_Tp>`:



#### 5.319.1 Detailed Description

**template<typename \_Tp> struct `std::_Reference_wrapper_base<_Tp>`**

Derives from [unary\\_function](#) or [binary\\_function](#) when it can. Specializations handle all of the easy cases. The primary template determines what to do with a class type, which may derive from both [unary\\_function](#) and [binary\\_function](#).

Definition at line 307 of file `functional`.

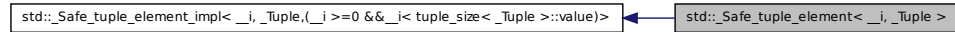
The documentation for this struct was generated from the following file:

- [functional](#)

## 5.320 `std::_Safe_tuple_element< __i, _Tuple >` Struct Template Reference 1561

### 5.320 `std::_Safe_tuple_element< __i, _Tuple >` Struct Template Reference

Inheritance diagram for `std::_Safe_tuple_element< __i, _Tuple >`:



#### 5.320.1 Detailed Description

`template<int __i, typename _Tuple> struct std::_Safe_tuple_element< __i, _Tuple >`

Like `tuple_element`, but returns `_No_tuple_element` when `tuple_element` would return an error.

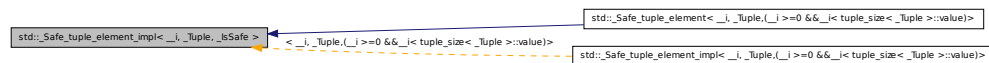
Definition at line 928 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

### 5.321 `std::_Safe_tuple_element_impl< __i, _Tuple, _IsSafe >` Struct Template Reference

Inheritance diagram for `std::_Safe_tuple_element_impl< __i, _Tuple, _IsSafe >`:



#### 5.321.1 Detailed Description

`template<int __i, typename _Tuple, bool _IsSafe> struct std::_Safe_tuple_element_impl< __i, _Tuple, _IsSafe >`

Implementation helper for [\\_Safe\\_tuple\\_element](#). This primary template handles the case where it is safe to use `tuple_element`.

### 5.322 `std::_Safe_tuple_element_impl< __i, _Tuple, false >` Struct Template Reference 1562

---

Definition at line 909 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

### 5.322 `std::_Safe_tuple_element_impl< __i, _Tuple, false >` Struct Template Reference

#### Public Types

- `typedef _No_tuple_element type`

#### 5.322.1 Detailed Description

`template<int __i, typename _Tuple> struct std::_Safe_tuple_element_impl< __i, _Tuple, false >`

Implementation helper for [\\_Safe\\_tuple\\_element](#). This partial specialization handles the case where it is not safe to use `tuple_element`. We just return `_No_tuple_element`.

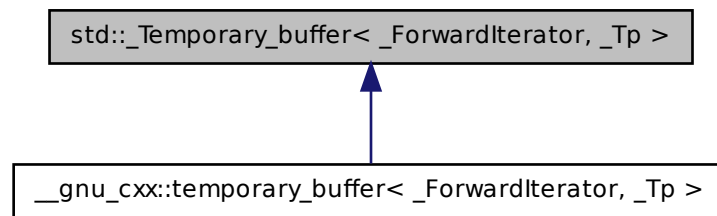
Definition at line 918 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

### 5.323 `std::_Temporary_buffer<_ForwardIterator, _Tp>` Class Template Reference

Inheritance diagram for `std::_Temporary_buffer<_ForwardIterator, _Tp>`:



#### Public Types

- typedef pointer **iterator**
- typedef value\_type \* **pointer**
- typedef ptrdiff\_t **size\_type**
- typedef \_Tp **value\_type**

#### Public Member Functions

- [\\_Temporary\\_buffer](#) (`_ForwardIterator __first, _ForwardIterator __last`)
- [iterator begin](#) ()
- [iterator end](#) ()
- size\_type [requested\\_size](#) () const
- size\_type [size](#) () const

#### Protected Attributes

- pointer **\_M\_buffer**
- size\_type **\_M\_len**
- size\_type **\_M\_original\_len**

### 5.323.1 Detailed Description

**template<typename \_ForwardIterator, typename \_Tp> class std::\_Temporary\_buffer<\_ForwardIterator, \_Tp>**

This class is used in two places: [stl\\_algo.h](#) and `ext/memory`, where it is wrapped as the `temporary_buffer` class. See `temporary_buffer` docs for more notes.

Definition at line 123 of file `stl_tempbuf.h`.

### 5.323.2 Constructor & Destructor Documentation

**5.323.2.1 template<typename \_ForwardIterator, typename \_Tp> std::\_Temporary\_buffer<\_ForwardIterator, \_Tp>::\_Temporary\_buffer ( \_ForwardIterator \_\_first, \_ForwardIterator \_\_last )**

Constructs a temporary buffer of a size somewhere between zero and the size of the given range.

Definition at line 246 of file `stl_tempbuf.h`.

References `std::pair<_T1, _T2>::first`, `std::get_temporary_buffer()`, `std::return_temporary_buffer()`, and `std::pair<_T1, _T2>::second`.

### 5.323.3 Member Function Documentation

**5.323.3.1 template<typename \_ForwardIterator, typename \_Tp> iterator std::\_Temporary\_buffer<\_ForwardIterator, \_Tp>::begin ( ) [inline]**

As per Table mumble.

Definition at line 152 of file `stl_tempbuf.h`.

Referenced by `std::inplace_merge()`, `std::stable_partition()`, and `std::stable_sort()`.

**5.323.3.2 template<typename \_ForwardIterator, typename \_Tp> iterator std::\_Temporary\_buffer<\_ForwardIterator, \_Tp>::end ( ) [inline]**

As per Table mumble.

Definition at line 157 of file `stl_tempbuf.h`.

**5.323.3.3** `template<typename _ForwardIterator, typename _Tp> size_type  
std::_Temporary_buffer<_ForwardIterator, _Tp>::requested_size ( ) const [inline]`

Returns the size requested by the constructor; may be `>size()`.

Definition at line 147 of file `stl_tempbuf.h`.

Referenced by `std::stable_partition()`.

**5.323.3.4** `template<typename _ForwardIterator, typename _Tp> size_type  
std::_Temporary_buffer<_ForwardIterator, _Tp>::size ( ) const [inline]`

As per Table mumble.

Definition at line 142 of file `stl_tempbuf.h`.

Referenced by `std::inplace_merge()`, `std::stable_partition()`, and `std::stable_sort()`.

The documentation for this class was generated from the following file:

- [stl\\_tempbuf.h](#)

## 5.324 `std::_Tuple_impl<_Idx>` Struct Template Reference

### Protected Member Functions

- `void _M_swap_impl (_Tuple_impl &)`

### 5.324.1 Detailed Description

`template<std::size_t _Idx> struct std::_Tuple_impl<_Idx>`

Zero-element tuple implementation. This is the basis case for the inheritance recursion.

Definition at line 132 of file `tuple`.

The documentation for this struct was generated from the following file:

- [tuple](#)



## 5.325 `std::_Tuple_impl<_Idx, _Head, _Tail...>` Struct Template Reference

Inherits `_Tuple_impl<_Idx+1, _Tail...>`, and `_Head_base<_Idx, _Head, std::is_empty<_Head>::value>`.

### Public Types

- typedef `_Head_base<_Idx, _Head, std::is\_empty<_Head>::value>` **\_Base**
- typedef `_Tuple_impl<_Idx+1, _Tail...>` **\_Inherited**

### Public Member Functions

- constexpr **\_Tuple\_impl** (const `_Tuple_impl` &)
- **\_Tuple\_impl** (`_Tuple_impl` &&\_\_in)
- template<typename... `_UElements`>  
**\_Tuple\_impl** (const `_Tuple_impl`<\_Idx, `_UElements`...> &\_\_in)
- template<typename... `_UElements`>  
**\_Tuple\_impl** (`_Tuple_impl`<\_Idx, `_UElements`...> &&\_\_in)
- constexpr **\_Tuple\_impl** (const `_Head` &\_\_head, const `_Tail` &...\_\_tail)
- template<typename `_UHead`, typename... `_UTail`>  
**\_Tuple\_impl** (`_UHead` &&\_\_head, `_UTail` &&...\_\_tail)
- `_Head` & **\_M\_head** ()
- const `_Head` & **\_M\_head** () const
- const `_Inherited` & **\_M\_tail** () const
- `_Inherited` & **\_M\_tail** ()
- template<typename... `_UElements`>  
`_Tuple_impl` & **operator=** (`_Tuple_impl`<\_Idx, `_UElements`...> &&\_\_in)
- `_Tuple_impl` & **operator=** (const `_Tuple_impl` &\_\_in)
- `_Tuple_impl` & **operator=** (`_Tuple_impl` &&\_\_in)
- template<typename... `_UElements`>  
`_Tuple_impl` & **operator=** (const `_Tuple_impl`<\_Idx, `_UElements`...> &\_\_in)

### Protected Member Functions

- void **\_M\_swap\_impl** (`_Tuple_impl` &\_\_in)

### 5.325.1 Detailed Description

template<std::size\_t \_Idx, typename \_Head, typename... \_Tail> struct std::\_Tuple\_impl< \_Idx, \_Head, \_Tail...>

Recursive tuple implementation. Here we store the `Head` element and derive from a `Tuple_impl` containing the remaining elements (which contains the `Tail`).

Definition at line 144 of file `tuple`.

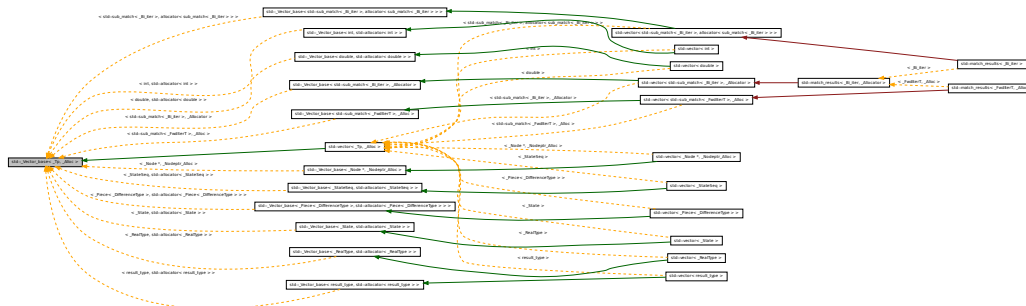
The documentation for this struct was generated from the following file:

- [tuple](#)

## 5.326 std::\_Vector\_base< \_Tp, \_Alloc > Struct Template Reference

See [bits/stl\\_deque.h](#)'s `_Deque_base` for an explanation.

Inheritance diagram for `std::_Vector_base< _Tp, _Alloc >`:



### Public Types

- typedef `_Alloc::template rebind< _Tp >::other` `_Tp_alloc_type`
- typedef `_Alloc` `allocator_type`

### Public Member Functions

- `_Vector_base` (const `allocator_type` &\_\_a)
- `_Vector_base` (`_Vector_base` &&\_\_x)
- `_Vector_base` (size\_t \_\_n)
- `_Vector_base` (size\_t \_\_n, const `allocator_type` &\_\_a)

- `_Tp_alloc_type::pointer _M_allocate (size_t __n)`
- `void _M_deallocate (typename _Tp_alloc_type::pointer __p, size_t __n)`
- `_Tp_alloc_type & _M_get_Tp_allocator ()`
- `const _Tp_alloc_type & _M_get_Tp_allocator () const`
- `allocator_type get_allocator () const`

### Public Attributes

- `_Vector_impl _M_impl`

#### 5.326.1 Detailed Description

`template<typename _Tp, typename _Alloc> struct std::_Vector_base<_Tp, _Alloc>`

See [bits/stl\\_deque.h](#)'s `_Deque_base` for an explanation.

Definition at line 71 of file `stl_vector.h`.

The documentation for this struct was generated from the following file:

- [stl\\_vector.h](#)

## 5.327 `std::_Weak_result_type<_Functor>` Struct Template Reference

Inheritance diagram for `std::_Weak_result_type<_Functor>`:



#### 5.327.1 Detailed Description

`template<typename _Functor> struct std::_Weak_result_type<_Functor>`

Strip top-level cv-qualifiers from the function object and let [\\_Weak\\_result\\_type\\_impl](#) perform the real work.

Definition at line 179 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

### 5.328 `std::Weak_result_type_impl<_Functor>` Struct Template Reference

The diagram illustrates the flow of information from a user-defined function to a compiled function. It starts with a function definition in a 'User-defined function' box, which is then transformed into a 'Compiled function' box. The process involves several steps, including the generation of a 'Compiled function' box and the final output of a 'Compiled function' box.

```
template<typename _Functor> struct std::_Weak_result_type_impl< _Functor
>
```

The documentation for this struct was generated from the following file:

- ### 5.329 `std::Weak_result_type_impl<_Res(&)(_ArgTypes...)>` Struct Template Reference

- `typedef _Res result_type`

```
template<typename _Res, typename... _ArgTypes> struct std::_Weak_result_
type_impl< _Res(&)(_ArgTypes...)>
```

The documentation for this struct was generated from the following file:

- 1569

### 5.330 `std::_Weak_result_type_impl< _Res(*)(_ArgTypes...)>` Struct Template Reference 1570

---

#### 5.330 `std::_Weak_result_type_impl< _Res(*)(_ArgTypes...)>` Struct Template Reference

Retrieve the result type for a function pointer.

##### Public Types

- `typedef _Res result_type`

##### 5.330.1 Detailed Description

`template<typename _Res, typename... _ArgTypes> struct std::_Weak_result_type_impl< _Res(*)(_ArgTypes...)>`

Retrieve the result type for a function pointer.

Definition at line 129 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

#### 5.331 `std::_Weak_result_type_impl< _Res(_ArgTypes...)>` Struct Template Reference

Retrieve the result type for a function type.

##### Public Types

- `typedef _Res result_type`

##### 5.331.1 Detailed Description

`template<typename _Res, typename... _ArgTypes> struct std::_Weak_result_type_impl< _Res(_ArgTypes...)>`

Retrieve the result type for a function type.

Definition at line 87 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

**5.332** `std::_Weak_result_type_impl<_Res(_Class::*)(_ArgTypes...) const >`  
**Struct Template Reference** 1571

---

**5.332** `std::_Weak_result_type_impl<_Res(_Class::*)(_ArgTypes...) const >` **Struct Template Reference**

Retrieve result type for a const member function pointer.

#### Public Types

- `typedef _Res result_type`

##### 5.332.1 Detailed Description

`template<typename _Res, typename _Class, typename... _ArgTypes> struct  
std::_Weak_result_type_impl<_Res(_Class::*)(_ArgTypes...) const >`

Retrieve result type for a const member function pointer.

Definition at line 147 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

**5.333** `std::_Weak_result_type_impl<_Res(_Class::*)(_ArgTypes...) const volatile >` **Struct Template Reference**

Retrieve result type for a const volatile member function pointer.

#### Public Types

- `typedef _Res result_type`

##### 5.333.1 Detailed Description

`template<typename _Res, typename _Class, typename... _ArgTypes> struct  
std::_Weak_result_type_impl<_Res(_Class::*)(_ArgTypes...) const volatile >`

Retrieve result type for a const volatile member function pointer.

Definition at line 165 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

**5.334 `std::_Weak_result_type_impl<_Res(_Class::*)(_ArgTypes...) volatile >` Struct Template Reference**

Retrieve result type for a volatile member function pointer.

**Public Types**

- `typedef _Res result_type`

**5.334.1 Detailed Description**

`template<typename _Res, typename _Class, typename... _ArgTypes> struct  
std::_Weak_result_type_impl<_Res(_Class::*)(_ArgTypes...) volatile >`

Retrieve result type for a volatile member function pointer.

Definition at line 156 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

**5.335 `std::_Weak_result_type_impl<_Res(_Class::*)(_ArgTypes...) >` Struct Template Reference**

Retrieve result type for a member function pointer.

**Public Types**

- `typedef _Res result_type`

**5.335.1 Detailed Description**

`template<typename _Res, typename _Class, typename... _ArgTypes> struct  
std::_Weak_result_type_impl<_Res(_Class::*)(_ArgTypes...) >`

Retrieve result type for a member function pointer.

Definition at line 138 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

### 5.336 `std::add_const<_Tp>` Struct Template Reference

[add\\_const](#)

#### Public Types

- typedef `_Tp const` **type**

#### 5.336.1 Detailed Description

`template<typename _Tp> struct std::add_const<_Tp>`

[add\\_const](#)

Definition at line 443 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.337 `std::add_cv<_Tp>` Struct Template Reference

[add\\_cv](#)

#### Public Types

- typedef `add_const<typename add_volatile<_Tp>::type>::type` **type**

#### 5.337.1 Detailed Description

`template<typename _Tp> struct std::add_cv<_Tp>`

[add\\_cv](#)

Definition at line 453 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.338 `std::add_lvalue_reference<_Tp>` Struct Template Reference

[add\\_lvalue\\_reference](#)



Inherits `std::__add_lvalue_reference_helper< _Tp >`.

#### Public Types

- `typedef _Tp type`

##### 5.338.1 Detailed Description

`template<typename _Tp> struct std::add_lvalue_reference< _Tp >`

[add\\_lvalue\\_reference](#)

Definition at line 571 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.339 `std::add_pointer< _Tp >` Struct Template Reference

[add\\_pointer](#)

#### Public Types

- `typedef remove\_reference< \_Tp >::type * type`

##### 5.339.1 Detailed Description

`template<typename _Tp> struct std::add_pointer< _Tp >`

[add\\_pointer](#)

Definition at line 508 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.340 `std::add_rvalue_reference< _Tp >` Struct Template Reference

[add\\_rvalue\\_reference](#)

Inherits `std::__add_rvalue_reference_helper< _Tp >`.

**Public Types**

- `typedef _Tp type`

**5.340.1 Detailed Description**

`template<typename _Tp> struct std::add_rvalue_reference< _Tp >`

[add\\_rvalue\\_reference](#)

Definition at line 586 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

**5.341 `std::add_volatile< _Tp >` Struct Template Reference**

[add\\_volatile](#)

**Public Types**

- `typedef _Tp volatile type`

**5.341.1 Detailed Description**

`template<typename _Tp> struct std::add_volatile< _Tp >`

[add\\_volatile](#)

Definition at line 448 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

**5.342 `std::adopt_lock_t` Struct Reference**

Assume the calling thread has already obtained mutex ownership /// and manage it.

**5.342.1 Detailed Description**

Assume the calling thread has already obtained mutex ownership /// and manage it.

Definition at line 429 of file `mutex`.

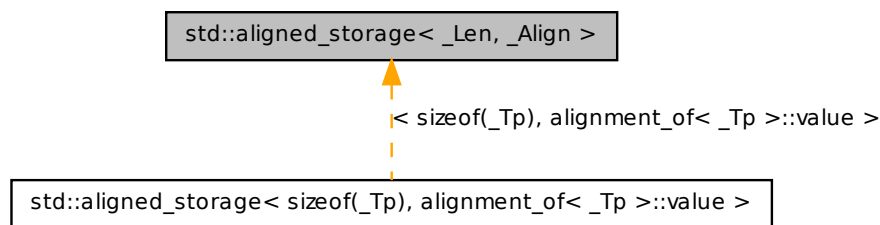
The documentation for this struct was generated from the following file:

- [mutex](#)

### 5.343 `std::aligned_storage< _Len, _Align >` Struct Template Reference

Alignment type.

Inheritance diagram for `std::aligned_storage< _Len, _Align >`:



#### 5.343.1 Detailed Description

```
template<std::size_t _Len, std::size_t _Align = __alignof__(typename __-
aligned_storage_msa<_Len>::__type)> struct std::aligned_storage< _Len, _-
Align >
```

Alignment type. The value of `_Align` is a default-alignment which shall be the most stringent alignment requirement for any C++ object type whose size is no greater than `_Len` (3.9). The member typedef type shall be a POD type suitable for use as uninitialized storage for any object whose size is at most `_Len` and whose alignment is a divisor of `_Align`.

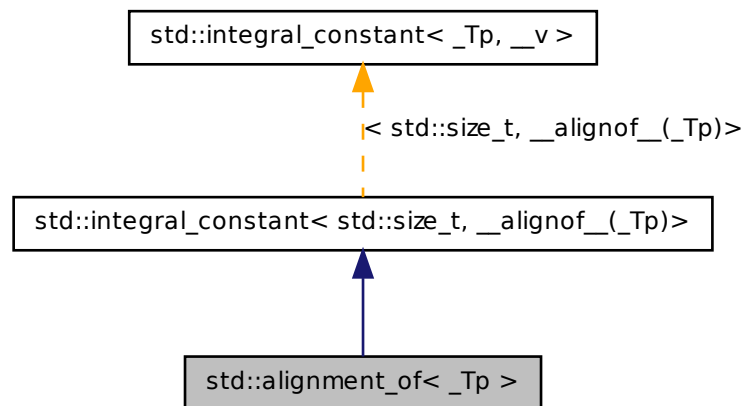
Definition at line 822 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

**5.344 std::alignment\_of< \_Tp > Struct Template Reference**[alignment\\_of](#)

Inheritance diagram for std::alignment\_of&lt; \_Tp &gt;:

**Public Types**

- typedef [integral\\_constant](#)< std::size\_t, \_\_v > **type**
- typedef std::size\_t **value\_type**

**Public Member Functions**

- constexpr **operator value\_type** ()

**Static Public Attributes**

- static constexpr std::size\_t **value**

### 5.344.1 Detailed Description

template<typename \_Tp> struct std::alignment\_of< \_Tp >

[alignment\\_of](#)

Definition at line 367 of file type\_traits.

The documentation for this struct was generated from the following file:

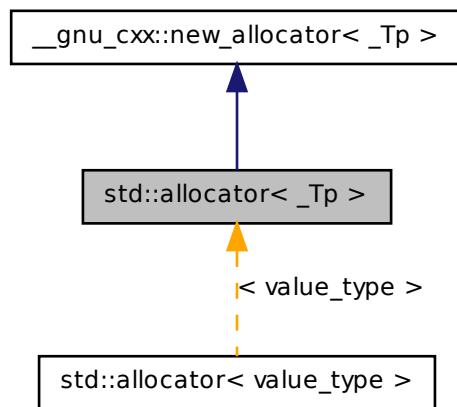
- [type\\_traits](#)

## 5.345 std::allocator< \_Tp > Class Template Reference

The *standard* allocator, as per [20.4].

Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt04ch11.html>.

Inheritance diagram for std::allocator< \_Tp >:



### Public Types

- typedef const \_Tp \* **const\_pointer**
- typedef const \_Tp & **const\_reference**
- typedef ptrdiff\_t **difference\_type**

- `typedef _Tp * pointer`
- `typedef _Tp & reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

### Public Member Functions

- **allocator** (const `allocator` &\_\_a) throw ()
- `template<typename _Tp1 >`  
**allocator** (const `allocator`< \_Tp1 > &) throw ()
- `pointer address` (reference \_\_x) const
- `const_pointer address` (const\_reference \_\_x) const
- `pointer allocate` (size\_type \_\_n, const void \*==0)
- `void construct` (pointer \_\_p, const \_Tp &\_\_val)
- `template<typename... _Args>`  
**void construct** (pointer \_\_p, \_Args &&...\_\_args)
- `void deallocate` (pointer \_\_p, size\_type)
- `void destroy` (pointer \_\_p)
- `size_type max_size` () const throw ()

#### 5.345.1 Detailed Description

`template<typename _Tp> class std::allocator< _Tp >`

The *standard* allocator, as per [20.4].

Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt04ch11.html>.

Definition at line 92 of file `allocator.h`.

The documentation for this class was generated from the following file:

- [allocator.h](#)

### 5.346 `std::allocator< void >` Class Template Reference

`allocator<void>` specialization.

### Public Types

- `typedef const void * const_pointer`
- `typedef ptrdiff_t difference_type`
- `typedef void * pointer`
- `typedef size_t size_type`
- `typedef void value_type`

### 5.346.1 Detailed Description

`template<> class std::allocator< void >`

[allocator<void>](#) specialization.

Definition at line 70 of file `allocator.h`.

The documentation for this class was generated from the following file:

- [allocator.h](#)

## 5.347 `std::allocator_arg_t` Struct Reference

[allocator.tag]

### 5.347.1 Detailed Description

[allocator.tag]

Definition at line 211 of file `allocator.h`.

The documentation for this struct was generated from the following file:

- [allocator.h](#)

## 5.348 `std::array< _Tp, _Nm >` Struct Template Reference

A standard container for storing a fixed size sequence of elements.

### Public Types

- `typedef const value_type * const_iterator`
- `typedef const _Tp * const_pointer`
- `typedef const value_type & const_reference`
- `typedef std::reverse\_iterator< const_iterator > const_reverse_iterator`
- `typedef std::ptrdiff_t difference_type`
- `typedef value_type * iterator`
- `typedef _Tp * pointer`
- `typedef value_type & reference`
- `typedef std::reverse\_iterator< iterator > reverse_iterator`
- `typedef std::size_t size_type`
- `typedef _Tp value_type`

**Public Member Functions**

- reference **at** (size\_type \_\_n)
- const\_reference **at** (size\_type \_\_n) const
- reference **back** ()
- const\_reference **back** () const
- [iterator](#) **begin** ()
- const\_iterator **begin** () const
- const\_iterator **cbegin** () const
- const\_iterator **end** () const
- [const\\_reverse\\_iterator](#) **crbegin** () const
- [const\\_reverse\\_iterator](#) **crend** () const
- const \_Tp \* **data** () const
- \_Tp \* **data** ()
- constexpr bool **empty** () const
- [iterator](#) **end** ()
- const\_iterator **end** () const
- void **fill** (const value\_type &\_\_u)
- reference **front** ()
- const\_reference **front** () const
- constexpr size\_type **max\_size** () const
- reference **operator**[ ] (size\_type \_\_n)
- const\_reference **operator**[ ] (size\_type \_\_n) const
- [const\\_reverse\\_iterator](#) **rbegin** () const
- [reverse\\_iterator](#) **rbegin** ()
- [reverse\\_iterator](#) **rend** ()
- [const\\_reverse\\_iterator](#) **rend** () const
- constexpr size\_type **size** () const
- void **swap** ([array](#) &\_\_other)

**Public Attributes**

- value\_type **\_M\_instance** [\_Nm?\_Nm:1]

**5.348.1 Detailed Description**

`template<typename _Tp, std::size_t _Nm> struct std::array<_Tp, _Nm>`

A standard container for storing a fixed size sequence of elements. Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#).

Sets support random access iterators.



**Parameters**

*Tp*  Type of element. Required to be a complete type.

*N*  Number of elements.

Definition at line 60 of file `array`.

The documentation for this struct was generated from the following file:

- [array](#)

**5.349 `std::atomic<_Tp>` Struct Template Reference**

`atomic` /// 29.4.3, Generic atomic type, primary class template.

**Public Member Functions**

- **atomic** (const [atomic](#) &)
- constexpr **atomic** (\_Tp \_\_i)
- bool **compare\_exchange\_strong** (\_Tp &, \_Tp, [memory\\_order](#), [memory\\_order](#))
- bool **compare\_exchange\_strong** (\_Tp &, \_Tp, [memory\\_order](#), [memory\\_order](#)) volatile
- bool **compare\_exchange\_strong** (\_Tp &, \_Tp, [memory\\_order](#)=[memory\\_order\\_seq\\_cst](#))
- bool **compare\_exchange\_strong** (\_Tp &, \_Tp, [memory\\_order](#)=[memory\\_order\\_seq\\_cst](#)) volatile
- bool **compare\_exchange\_weak** (\_Tp &, \_Tp, [memory\\_order](#)=[memory\\_order\\_seq\\_cst](#)) volatile
- bool **compare\_exchange\_weak** (\_Tp &, \_Tp, [memory\\_order](#), [memory\\_order](#))
- bool **compare\_exchange\_weak** (\_Tp &, \_Tp, [memory\\_order](#)=[memory\\_order\\_seq\\_cst](#))
- bool **compare\_exchange\_weak** (\_Tp &, \_Tp, [memory\\_order](#), [memory\\_order](#)) volatile
- \_Tp **exchange** (\_Tp \_\_i, [memory\\_order](#)=[memory\\_order\\_seq\\_cst](#)) volatile
- \_Tp **exchange** (\_Tp \_\_i, [memory\\_order](#)=[memory\\_order\\_seq\\_cst](#))
- bool **is\_lock\_free** () const
- bool **is\_lock\_free** () const volatile
- \_Tp **load** ([memory\\_order](#)=[memory\\_order\\_seq\\_cst](#)) const volatile
- \_Tp **load** ([memory\\_order](#)=[memory\\_order\\_seq\\_cst](#)) const
- **operator \_Tp** () const
- **operator \_Tp** () const volatile
- [atomic](#) & **operator=** (const [atomic](#) &) volatile

- `_Tp operator= (_Tp __i)`
- `atomic & operator= (const atomic &)`
- `_Tp operator= (_Tp __i) volatile`
- `void store (_Tp, memory_order=memory_order_seq_cst)`
- `void store (_Tp, memory_order=memory_order_seq_cst) volatile`

### 5.349.1 Detailed Description

`template<typename _Tp> struct std::atomic<_Tp>`

`atomic` /// 29.4.3, Generic atomic type, primary class template.

Definition at line 155 of file `atomic`.

The documentation for this struct was generated from the following file:

- `atomic`

## 5.350 `std::atomic<_Tp*>` Struct Template Reference

Partial specialization for pointer types.

Inherits `__atomic0::atomic_address`.

### Public Member Functions

- `constexpr atomic (_Tp *__v)`
- `atomic (const atomic &)`
- `bool compare_exchange_strong (_Tp *&, _Tp *, memory_order, memory_order)`
- `bool compare_exchange_strong (_Tp *&, _Tp *, memory_order, memory_order) volatile`
- `bool compare_exchange_strong (_Tp *&, _Tp *, memory_order=memory_order_seq_cst)`
- `bool compare_exchange_strong (_Tp *&, _Tp *, memory_order=memory_order_seq_cst) volatile`
- `bool compare_exchange_weak (_Tp *&, _Tp *, memory_order, memory_order)`
- `bool compare_exchange_weak (_Tp *&, _Tp *, memory_order, memory_order) volatile`
- `bool compare_exchange_weak (_Tp *&, _Tp *, memory_order=memory_order_seq_cst)`
- `bool compare_exchange_weak (_Tp *&, _Tp *, memory_order=memory_order_seq_cst) volatile`

- `_Tp * exchange` (`_Tp *`, `memory_order=memory_order_seq_cst`)
- `_Tp * exchange` (`_Tp *`, `memory_order=memory_order_seq_cst`) volatile
- `_Tp * fetch_add` (`ptrdiff_t`, `memory_order=memory_order_seq_cst`)
- `_Tp * fetch_add` (`ptrdiff_t`, `memory_order=memory_order_seq_cst`) volatile
- `_Tp * fetch_sub` (`ptrdiff_t`, `memory_order=memory_order_seq_cst`)
- `_Tp * fetch_sub` (`ptrdiff_t`, `memory_order=memory_order_seq_cst`) volatile
- `_Tp * load` (`memory_order=memory_order_seq_cst`) const
- `_Tp * load` (`memory_order=memory_order_seq_cst`) const volatile
- `operator _Tp *` () const
- `operator _Tp *` () const volatile
- `_Tp * operator++` ()
- `_Tp * operator++` () volatile
- `_Tp * operator++` (int)
- `_Tp * operator++` (int) volatile
- `_Tp * operator+=` (`ptrdiff_t __d`) volatile
- `_Tp * operator+=` (`ptrdiff_t __d`)
- `_Tp * operator--` () volatile
- `_Tp * operator--` ()
- `_Tp * operator--` (int)
- `_Tp * operator--` (int) volatile
- `_Tp * operator-=` (`ptrdiff_t __d`) volatile
- `_Tp * operator-=` (`ptrdiff_t __d`)
- `_Tp * operator=` (`_Tp *__v`)
- `atomic & operator=` (const `atomic &`) volatile
- `_Tp * operator=` (`_Tp *__v`) volatile
- void `store` (`_Tp *`, `memory_order=memory_order_seq_cst`)
- void `store` (`_Tp *`, `memory_order=memory_order_seq_cst`) volatile

### 5.350.1 Detailed Description

`template<typename _Tp> struct std::atomic<_Tp*>`

Partial specialization for pointer types.

Definition at line 233 of file `atomic`.

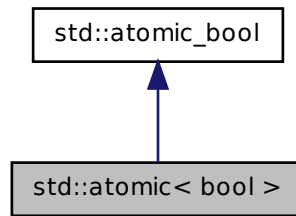
The documentation for this struct was generated from the following file:

- `atomic`

### 5.351 `std::atomic< bool >` Struct Template Reference

Explicit specialization for `bool`.

Inheritance diagram for `std::atomic< bool >`:



#### Public Types

- typedef `atomic_bool` `__base_type`
- typedef `bool` `__integral_type`

#### Public Member Functions

- `atomic` (const `atomic` &)
- constexpr `atomic` (`__integral_type` \_\_i)
- `bool compare_exchange_strong` (`bool` &\_\_i1, `bool` \_\_i2, `memory_order` \_\_m1, `memory_order` \_\_m2)
- `bool compare_exchange_strong` (`bool` &\_\_i1, `bool` \_\_i2, `memory_order` \_\_m1, `memory_order` \_\_m2) volatile
- `bool compare_exchange_strong` (`bool` &\_\_i1, `bool` \_\_i2, `memory_order` \_\_m=memory\_order\_seq\_cst)
- `bool compare_exchange_strong` (`bool` &\_\_i1, `bool` \_\_i2, `memory_order` \_\_m=memory\_order\_seq\_cst) volatile
- `bool compare_exchange_weak` (`bool` &\_\_i1, `bool` \_\_i2, `memory_order` \_\_m1, `memory_order` \_\_m2)
- `bool compare_exchange_weak` (`bool` &\_\_i1, `bool` \_\_i2, `memory_order` \_\_m1, `memory_order` \_\_m2) volatile
- `bool compare_exchange_weak` (`bool` &\_\_i1, `bool` \_\_i2, `memory_order` \_\_m=memory\_order\_seq\_cst)

- `bool compare_exchange_weak (bool &__i1, bool __i2, memory\_order __m=memory_order_seq_cst) volatile`
- `bool exchange (bool __i, memory\_order __m=memory_order_seq_cst) volatile`
- `bool exchange (bool __i, memory\_order __m=memory_order_seq_cst)`
- `bool is_lock_free () const`
- `bool is_lock_free () const volatile`
- `bool load (memory\_order __m=memory_order_seq_cst) const`
- `bool load (memory\_order __m=memory_order_seq_cst) const volatile`
- `operator bool () const`
- `operator bool () const volatile`
- `atomic & operator= (const atomic &)`
- `atomic & operator= (const atomic &) volatile`
- `void store (bool __i, memory\_order __m=memory_order_seq_cst) volatile`
- `void store (bool __i, memory\_order __m=memory_order_seq_cst)`

### 5.351.1 Detailed Description

`template<> struct std::atomic< bool >`

Explicit specialization for `bool`.

Definition at line 361 of file `atomic`.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 5.352 `std::atomic< char >` Struct Template Reference

Explicit specialization for `char`.

Inherits `__atomic0::__atomic_base< char >`.

### Public Types

- `typedef atomic\_char __base_type`
- `typedef char __integral_type`

### Public Member Functions

- `constexpr atomic (__integral_type __i)`
- `atomic (const atomic &)`
- `atomic & operator= (const atomic &) volatile`
- `atomic & operator= (const atomic &)`

### 5.352.1 Detailed Description

**template<> struct std::atomic< char >**

Explicit specialization for char.

Definition at line 380 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 5.353 `std::atomic< char16_t >` Struct Template Reference

Explicit specialization for char16\_t.

Inherits `__atomic0::__atomic_base< char16_t >`.

### Public Types

- typedef [atomic\\_char16\\_t](#) **\_\_base\_type**
- typedef char16\_t **\_\_integral\_type**

### Public Member Functions

- constexpr **atomic** (`__integral_type __i`)
- **atomic** (const [atomic](#) &)
- [atomic](#) & **operator=** (const [atomic](#) &) volatile
- [atomic](#) & **operator=** (const [atomic](#) &)

### 5.353.1 Detailed Description

**template<> struct std::atomic< char16\_t >**

Explicit specialization for char16\_t.

Definition at line 608 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

### 5.354 `std::atomic< char32_t >` Struct Template Reference

Explicit specialization for `char32_t`.

Inherits `__atomic0::__atomic_base< char32_t >`.

#### Public Types

- typedef `atomic_char32_t __base_type`
- typedef `char32_t __integral_type`

#### Public Member Functions

- constexpr `atomic` (`__integral_type __i`)
- `atomic` (const `atomic` &)
- `atomic` & `operator=` (const `atomic` &) volatile
- `atomic` & `operator=` (const `atomic` &)

#### 5.354.1 Detailed Description

`template<> struct std::atomic< char32_t >`

Explicit specialization for `char32_t`.

Definition at line 627 of file `atomic`.

The documentation for this struct was generated from the following file:

- `atomic`

### 5.355 `std::atomic< int >` Struct Template Reference

Explicit specialization for `int`.

Inherits `__atomic0::__atomic_base< int >`.

#### Public Types

- typedef `atomic_int __base_type`
- typedef `int __integral_type`

### Public Member Functions

- constexpr **atomic** (\_\_integral\_type \_\_i)
- **atomic** (const [atomic](#) &)
- [atomic](#) & **operator=** (const [atomic](#) &) volatile
- [atomic](#) & **operator=** (const [atomic](#) &)

#### 5.355.1 Detailed Description

**template<> struct std::atomic< int >**

Explicit specialization for int.

Definition at line 475 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 5.356 `std::atomic< long >` Struct Template Reference

Explicit specialization for long.

Inherits `__atomic0::__atomic_base< long >`.

### Public Types

- typedef [atomic\\_long](#) **\_\_base\_type**
- typedef long **\_\_integral\_type**

### Public Member Functions

- constexpr **atomic** (\_\_integral\_type \_\_i)
- **atomic** (const [atomic](#) &)
- [atomic](#) & **operator=** (const [atomic](#) &) volatile
- [atomic](#) & **operator=** (const [atomic](#) &)

#### 5.356.1 Detailed Description

**template<> struct std::atomic< long >**

Explicit specialization for long.

Definition at line 513 of file atomic.



The documentation for this struct was generated from the following file:

- [atomic](#)

### 5.357 `std::atomic< long long >` Struct Template Reference

Explicit specialization for long long.

Inherits `__atomic0::__atomic_base< long long >`.

#### Public Types

- typedef [atomic\\_llong](#) `__base_type`
- typedef long long `__integral_type`

#### Public Member Functions

- constexpr `atomic` (`__integral_type __i`)
- `atomic` (const [atomic](#) &)
- [atomic](#) & `operator=` (const [atomic](#) &) volatile
- [atomic](#) & `operator=` (const [atomic](#) &)

#### 5.357.1 Detailed Description

`template<> struct std::atomic< long long >`

Explicit specialization for long long.

Definition at line 551 of file `atomic`.

The documentation for this struct was generated from the following file:

- [atomic](#)

### 5.358 `std::atomic< short >` Struct Template Reference

Explicit specialization for short.

Inherits `__atomic0::__atomic_base< short >`.

#### Public Types

- typedef [atomic\\_short](#) `__base_type`
- typedef short `__integral_type`

### Public Member Functions

- constexpr **atomic** (\_\_integral\_type \_\_i)
- **atomic** (const [atomic](#) &)
- [atomic](#) & **operator=** (const [atomic](#) &) volatile
- [atomic](#) & **operator=** (const [atomic](#) &)

#### 5.358.1 Detailed Description

**template<> struct std::atomic< short >**

Explicit specialization for short.

Definition at line 437 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

### 5.359 `std::atomic< signed char >` Struct Template Reference

Explicit specialization for signed char.

Inherits `__atomic0::__atomic_base< signed char >`.

### Public Types

- typedef [atomic\\_schar](#) **\_\_base\_type**
- typedef signed char **\_\_integral\_type**

### Public Member Functions

- constexpr **atomic** (\_\_integral\_type \_\_i)
- **atomic** (const [atomic](#) &)
- [atomic](#) & **operator=** (const [atomic](#) &) volatile
- [atomic](#) & **operator=** (const [atomic](#) &)

#### 5.359.1 Detailed Description

**template<> struct std::atomic< signed char >**

Explicit specialization for signed char.

Definition at line 399 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

### 5.360 `std::atomic< unsigned char >` Struct Template Reference

Explicit specialization for unsigned char.

Inherits `__atomic0::__atomic_base< unsigned char >`.

#### Public Types

- typedef [atomic\\_uchar](#) `__base_type`
- typedef unsigned char `__integral_type`

#### Public Member Functions

- constexpr `atomic` (`__integral_type __i`)
- `atomic` (const [atomic](#) &)
- [atomic](#) & `operator=` (const [atomic](#) &) volatile
- [atomic](#) & `operator=` (const [atomic](#) &)

#### 5.360.1 Detailed Description

`template<> struct std::atomic< unsigned char >`

Explicit specialization for unsigned char.

Definition at line 418 of file `atomic`.

The documentation for this struct was generated from the following file:

- [atomic](#)

### 5.361 `std::atomic< unsigned int >` Struct Template Reference

Explicit specialization for unsigned int.

Inherits `__atomic0::__atomic_base< unsigned int >`.

#### Public Types

- typedef [atomic\\_uint](#) `__base_type`
- typedef unsigned int `__integral_type`

### Public Member Functions

- constexpr **atomic** (\_\_integral\_type \_\_i)
- **atomic** (const [atomic](#) &)
- [atomic](#) & **operator=** (const [atomic](#) &) volatile
- [atomic](#) & **operator=** (const [atomic](#) &)

#### 5.361.1 Detailed Description

**template<> struct std::atomic< unsigned int >**

Explicit specialization for unsigned int.

Definition at line 494 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 5.362 `std::atomic< unsigned long >` Struct Template Reference

Explicit specialization for unsigned long.

Inherits `__atomic0::__atomic_base< unsigned long >`.

### Public Types

- typedef [atomic\\_ulong](#) **\_\_base\_type**
- typedef unsigned long **\_\_integral\_type**

### Public Member Functions

- constexpr **atomic** (\_\_integral\_type \_\_i)
- **atomic** (const [atomic](#) &)
- [atomic](#) & **operator=** (const [atomic](#) &) volatile
- [atomic](#) & **operator=** (const [atomic](#) &)

#### 5.362.1 Detailed Description

**template<> struct std::atomic< unsigned long >**

Explicit specialization for unsigned long.

Definition at line 532 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

### 5.363 `std::atomic< unsigned long long >` Struct Template Reference

Explicit specialization for unsigned long long.

Inherits `__atomic0::__atomic_base< unsigned long long >`.

#### Public Types

- typedef [atomic\\_ulong](#) `__base_type`
- typedef unsigned long long `__integral_type`

#### Public Member Functions

- constexpr **atomic** (`__integral_type __i`)
- **atomic** (const [atomic](#) &)
- [atomic](#) & **operator=** (const [atomic](#) &) volatile
- [atomic](#) & **operator=** (const [atomic](#) &)

#### 5.363.1 Detailed Description

**template<> struct std::atomic< unsigned long long >**

Explicit specialization for unsigned long long.

Definition at line 570 of file `atomic`.

The documentation for this struct was generated from the following file:

- [atomic](#)

### 5.364 `std::atomic< unsigned short >` Struct Template Reference

Explicit specialization for unsigned short.

Inherits `__atomic0::__atomic_base< unsigned short >`.

### Public Types

- typedef `atomic_ushort` `__base_type`
- typedef unsigned short `__integral_type`

### Public Member Functions

- constexpr `atomic` (`__integral_type __i`)
- `atomic` (const `atomic` &)
- `atomic` & `operator=` (const `atomic` &) volatile
- `atomic` & `operator=` (const `atomic` &)

#### 5.364.1 Detailed Description

`template<> struct std::atomic< unsigned short >`

Explicit specialization for unsigned short.

Definition at line 456 of file `atomic`.

The documentation for this struct was generated from the following file:

- `atomic`

## 5.365 `std::atomic< wchar_t >` Struct Template Reference

Explicit specialization for `wchar_t`.

Inherits `__atomic0::__atomic_base< wchar_t >`.

### Public Types

- typedef `atomic_wchar_t` `__base_type`
- typedef `wchar_t` `__integral_type`

### Public Member Functions

- constexpr `atomic` (`__integral_type __i`)
- `atomic` (const `atomic` &)
- `atomic` & `operator=` (const `atomic` &) volatile
- `atomic` & `operator=` (const `atomic` &)

### 5.365.1 Detailed Description

**template<> struct std::atomic< wchar\_t >**

Explicit specialization for wchar\_t.

Definition at line 589 of file atomic.

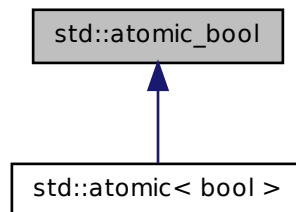
The documentation for this struct was generated from the following file:

- [atomic](#)

## 5.366 std::atomic\_bool Struct Reference

[atomic\\_bool](#)

Inheritance diagram for std::atomic\_bool:



### Public Member Functions

- **atomic\_bool** (const [atomic\\_bool](#) &)
- constexpr **atomic\_bool** (bool \_\_i)
- bool **compare\_exchange\_strong** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2)
- bool **compare\_exchange\_strong** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile
- bool **compare\_exchange\_strong** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile
- bool **compare\_exchange\_strong** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst)

- bool **compare\_exchange\_weak** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2)
- bool **compare\_exchange\_weak** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile
- bool **compare\_exchange\_weak** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile
- bool **compare\_exchange\_weak** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst)
- bool **exchange** (bool \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile
- bool **exchange** (bool \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst)
- bool **is\_lock\_free** () const volatile
- bool **is\_lock\_free** () const
- bool **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const volatile
- bool **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const
- **operator bool** () const volatile
- **operator bool** () const
- [atomic\\_bool](#) & **operator=** (const [atomic\\_bool](#) &) volatile
- [atomic\\_bool](#) & **operator=** (const [atomic\\_bool](#) &)
- bool **operator=** (bool \_\_i)
- void **store** (bool \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile
- void **store** (bool \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst)

### 5.366.1 Detailed Description

#### [atomic\\_bool](#)

Definition at line 56 of file `atomic`.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 5.367 std::auto\_ptr< \_Tp > Class Template Reference

A simple smart pointer providing strict ownership semantics.

### Public Types

- typedef `_Tp` [element\\_type](#)



**Public Member Functions**

- `auto_ptr` (`element_type` \*\_\_p=0) throw ()
- `auto_ptr` (`auto_ptr` &\_\_a) throw ()
- `template<typename _Tp1 >`  
`auto_ptr` (`auto_ptr`<\_Tp1> &\_\_a) throw ()
- `auto_ptr` (`auto_ptr_ref`< `element_type` > \_\_ref) throw ()
- `~auto_ptr` ()
- `element_type` \* `get` () const throw ()
- `template<typename _Tp1 >`  
`operator auto_ptr`<\_Tp1> () throw ()
- `template<typename _Tp1 >`  
`operator auto_ptr_ref`<\_Tp1> () throw ()
- `element_type` & `operator*` () const throw ()
- `element_type` \* `operator->` () const throw ()
- `auto_ptr` & `operator=` (`auto_ptr` &\_\_a) throw ()
- `template<typename _Tp1 >`  
`auto_ptr` & `operator=` (`auto_ptr`<\_Tp1> &\_\_a) throw ()
- `auto_ptr` & `operator=` (`auto_ptr_ref`< `element_type` > \_\_ref) throw ()
- `element_type` \* `release` () throw ()
- void `reset` (`element_type` \*\_\_p=0) throw ()

**5.367.1 Detailed Description**

`template<typename _Tp> class std::auto_ptr<_Tp>`

A simple smart pointer providing strict ownership semantics. The Standard says:

An `auto_ptr` owns the object it holds a pointer to. Copying an `auto_ptr` copies the pointer and transfers ownership to the destination. If more than one `auto_ptr` owns the same object at the same time the behavior of the program is undefined.

The uses of `auto_ptr` include providing temporary exception-safety for dynamically allocated memory, passing ownership of dynamically allocated memory to a function, and returning dynamically allocated memory from a function. `auto_ptr` does not meet the Container requirements for Standard Library `container` elements and thus instantiating a Standard Library container with an `auto_ptr` results in undefined behavior.

Quoted from [20.4.5]/3.

Good examples of what can and cannot be done with `auto_ptr` can be found in the libstdc++ testsuite.

\_GLIBCXX\_RESOLVE\_LIB\_DEFECTS 127. auto\_ptr<> conversion issues These resolutions have all been incorporated.

Definition at line 87 of file auto\_ptr.h.

### 5.367.2 Member Typedef Documentation

#### 5.367.2.1 template<typename \_Tp> typedef \_Tp std::auto\_ptr< \_Tp >::element\_type

The pointed-to type.

Definition at line 94 of file auto\_ptr.h.

### 5.367.3 Constructor & Destructor Documentation

#### 5.367.3.1 template<typename \_Tp> std::auto\_ptr< \_Tp >::auto\_ptr ( element\_type \* \_\_p = 0 ) throw () [inline, explicit]

An auto\_ptr is usually constructed from a raw pointer.

##### Parameters

*p* A pointer (defaults to NULL).

This object now *owns* the object pointed to by *p*.

Definition at line 103 of file auto\_ptr.h.

#### 5.367.3.2 template<typename \_Tp> std::auto\_ptr< \_Tp >::auto\_ptr ( auto\_ptr< \_Tp > & \_\_a ) throw () [inline]

An auto\_ptr can be constructed from another auto\_ptr.

##### Parameters

*a* Another auto\_ptr of the same type.

This object now *owns* the object previously owned by *a*, which has given up ownership.

Definition at line 112 of file auto\_ptr.h.

```
5.367.3.3 template<typename _Tp> template<typename _Tp1 >
 std::auto_ptr<_Tp>::auto_ptr (auto_ptr<_Tp1> & __a) throw
 () [inline]
```

An auto\_ptr can be constructed from another auto\_ptr.

#### Parameters

*a* Another auto\_ptr of a different but related type.

A pointer-to-Tp1 must be convertible to a pointer-to-Tp/element\_type.

This object now *owns* the object previously owned by *a*, which has given up ownership.

Definition at line 125 of file auto\_ptr.h.

```
5.367.3.4 template<typename _Tp> std::auto_ptr<_Tp>::~~auto_ptr ()
 [inline]
```

When the auto\_ptr goes out of scope, the object it owns is deleted. If it no longer owns anything (i.e., `get ()` is NULL), then this has no effect.

The C++ standard says there is supposed to be an empty throw specification here, but omitting it is standard conforming. Its presence can be detected only if `_Tp::~~_Tp()` throws, but this is prohibited. [17.4.3.6]/2

Definition at line 170 of file auto\_ptr.h.

```
5.367.3.5 template<typename _Tp> std::auto_ptr<_Tp>::auto_ptr (
 auto_ptr_ref<element_type> & __ref) throw () [inline]
```

Automatic conversions.

These operations convert an auto\_ptr into and from an `auto_ptr_ref` automatically as needed. This allows constructs such as

```
auto_ptr<Derived> func_returning_auto_ptr(...);
...
auto_ptr<Base> ptr = func_returning_auto_ptr(...);
```

Definition at line 260 of file auto\_ptr.h.

#### 5.367.4 Member Function Documentation

##### 5.367.4.1 `template<typename _Tp> element_type* std::auto_ptr< _Tp >::get ( void ) const throw () [inline]`

Bypassing the smart pointer.

#### Returns

The raw pointer being managed.

You can get a copy of the pointer that this object owns, for situations such as passing to a function which only accepts a raw pointer.

#### Note

This `auto_ptr` still owns the memory.

Definition at line 211 of file `auto_ptr.h`.

##### 5.367.4.2 `template<typename _Tp> element_type& std::auto_ptr< _Tp >::operator* ( ) const throw () [inline]`

Smart pointer dereferencing.

If this `auto_ptr` no longer owns anything, then this operation will crash. (For a smart pointer, *no longer owns anything* is the same as being a null pointer, and you know what happens when you dereference one of those...)

Definition at line 181 of file `auto_ptr.h`.

##### 5.367.4.3 `template<typename _Tp> element_type* std::auto_ptr< _Tp >::operator-> ( ) const throw () [inline]`

Smart pointer dereferencing.

This returns the pointer itself, which the language then will automatically cause to be dereferenced.

Definition at line 194 of file `auto_ptr.h`.

**5.367.4.4** `template<typename _Tp> template<typename _Tp1 > auto_ptr&  
std::auto_ptr< _Tp >::operator= ( auto_ptr< _Tp1 > & __a )  
throw () [inline]`

auto\_ptr assignment operator.

#### Parameters

*a* Another auto\_ptr of a different but related type.

A pointer-to-Tp1 must be convertible to a pointer-to-Tp/element\_type.

This object now *owns* the object previously owned by *a*, which has given up ownership. The object that this one *used* to own and track has been deleted.

Definition at line 154 of file auto\_ptr.h.

References std::auto\_ptr< \_Tp >::reset().

**5.367.4.5** `template<typename _Tp> auto_ptr& std::auto_ptr< _Tp  
>::operator= ( auto_ptr< _Tp > & __a ) throw () [inline]`

auto\_ptr assignment operator.

#### Parameters

*a* Another auto\_ptr of the same type.

This object now *owns* the object previously owned by *a*, which has given up ownership. The object that this one *used* to own and track has been deleted.

Definition at line 136 of file auto\_ptr.h.

References std::auto\_ptr< \_Tp >::reset().

**5.367.4.6** `template<typename _Tp> element_type* std::auto_ptr< _Tp  
>::release ( ) throw () [inline]`

Bypassing the smart pointer.

#### Returns

The raw pointer being managed.

You can get a copy of the pointer that this object owns, for situations such as passing to a function which only accepts a raw pointer.

#### Note

This `auto_ptr` no longer owns the memory. When this object goes out of scope, nothing will happen.

Definition at line 225 of file `auto_ptr.h`.

**5.367.4.7** `template<typename _Tp> void std::auto_ptr< _Tp >::reset ( element_type * __p = 0 ) throw () [inline]`

Forcibly deletes the managed object.

#### Parameters

*p* A pointer (defaults to NULL).

This object now *owns* the object pointed to by *p*. The previous object has been deleted.

Definition at line 240 of file `auto_ptr.h`.

Referenced by `std::auto_ptr< _Tp >::operator=()`.

The documentation for this class was generated from the following file:

- [auto\\_ptr.h](#)

## 5.368 `std::auto_ptr_ref< _Tp1 >` Struct Template Reference

### Public Member Functions

- `auto_ptr_ref ( _Tp1 * __p )`

### Public Attributes

- `_Tp1 * _M_ptr`

### 5.368.1 Detailed Description

`template<typename _Tp1> struct std::auto_ptr_ref< _Tp1 >`

A wrapper class to provide [auto\\_ptr](#) with reference semantics. For example, an [auto\\_ptr](#) can be assigned (or constructed from) the result of a function which returns an [auto\\_ptr](#)

### 5.369 `std::back_insert_iterator< _Container >` Class Template Reference 1604

by value.

All the [auto\\_ptr\\_ref](#) stuff should happen behind the scenes.

Definition at line 48 of file `auto_ptr.h`.

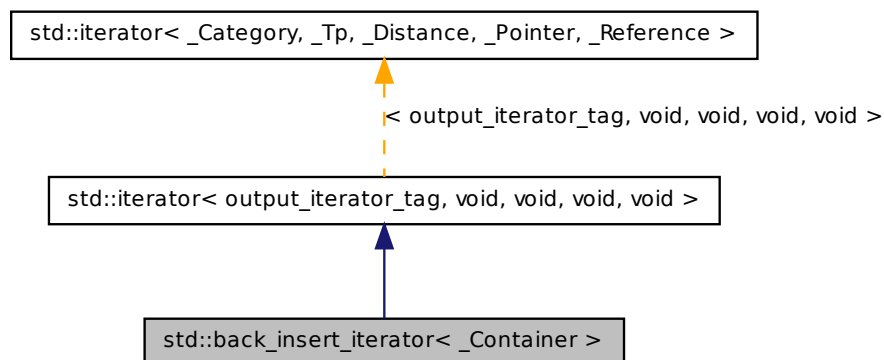
The documentation for this struct was generated from the following file:

- [auto\\_ptr.h](#)

### 5.369 `std::back_insert_iterator< _Container >` Class Template Reference

Turns assignment into insertion.

Inheritance diagram for `std::back_insert_iterator< _Container >`:



#### Public Types

- typedef `_Container` [container\\_type](#)
- typedef void [difference\\_type](#)
- typedef [output\\_iterator\\_tag](#) [iterator\\_category](#)
- typedef void [pointer](#)
- typedef void [reference](#)
- typedef void [value\\_type](#)

### Public Member Functions

- `back_insert_iterator` (`_Container &__x`)
- `back_insert_iterator` & `operator*` ()
- `back_insert_iterator` & `operator++` ()
- `back_insert_iterator` `operator++` (int)
- `back_insert_iterator` & `operator=` (const typename `_Container::value_type` &\_\_value)
- `back_insert_iterator` & `operator=` (typename `_Container::value_type` &&\_\_value)

### Protected Attributes

- `_Container * container`

#### 5.369.1 Detailed Description

**template<typename \_Container> class `std::back_insert_iterator<_Container>`**

Turns assignment into insertion. These are output iterators, constructed from a container-of-T. Assigning a T to the iterator appends it to the container using `push_back`.

Tip: Using the `back_inserter` function to create these iterators can save typing.

Definition at line 396 of file `stl_iterator.h`.

#### 5.369.2 Member Typedef Documentation

**5.369.2.1** **template<typename \_Container> typedef `_Container std::back_insert_iterator<_Container>::container_type`**

A nested typedef for the type of whatever container you used.

Definition at line 404 of file `stl_iterator.h`.

**5.369.2.2** **typedef void `std::iterator< output_iterator_tag, void, void, void, void>::difference_type` [inherited]**

Distance between iterators is represented as this type.

Definition at line 126 of file `stl_iterator_base_types.h`.



## **5.369 std::back\_insert\_iterator< \_Container > Class Template Reference 1606**

---

**5.369.2.3** `typedef output_iterator_tag std::iterator< output_iterator_tag , void , void , void , void >::iterator_category [inherited]`

One of the [tag types](#).

Definition at line 122 of file `stl_iterator_base_types.h`.

**5.369.2.4** `typedef void std::iterator< output_iterator_tag , void , void , void , void >::pointer [inherited]`

This type represents a pointer-to-value\_type.

Definition at line 128 of file `stl_iterator_base_types.h`.

**5.369.2.5** `typedef void std::iterator< output_iterator_tag , void , void , void , void >::reference [inherited]`

This type represents a reference-to-value\_type.

Definition at line 130 of file `stl_iterator_base_types.h`.

**5.369.2.6** `typedef void std::iterator< output_iterator_tag , void , void , void , void >::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 124 of file `stl_iterator_base_types.h`.

### **5.369.3 Constructor & Destructor Documentation**

**5.369.3.1** `template<typename _Container > std::back_insert_iterator< _Container >::back_insert_iterator ( _Container & __x ) [inline, explicit]`

The only way to create this iterator is with a container.

Definition at line 408 of file `stl_iterator.h`.

#### 5.369.4 Member Function Documentation

**5.369.4.1** `template<typename _Container> back_insert_iterator&  
std::back_insert_iterator<_Container>::operator* ( )  
[inline]`

Simply returns \*this.

Definition at line 446 of file stl\_iterator.h.

**5.369.4.2** `template<typename _Container> back_insert_iterator  
std::back_insert_iterator<_Container>::operator++ ( int )  
[inline]`

Simply returns \*this. (This iterator does not *move*.).

Definition at line 456 of file stl\_iterator.h.

**5.369.4.3** `template<typename _Container> back_insert_iterator&  
std::back_insert_iterator<_Container>::operator++ ( )  
[inline]`

Simply returns \*this. (This iterator does not *move*.).

Definition at line 451 of file stl\_iterator.h.

**5.369.4.4** `template<typename _Container> back_insert_iterator&  
std::back_insert_iterator<_Container>::operator= ( const  
typename _Container::value_type & __value ) [inline]`

#### Parameters

*value* An instance of whatever type `container_type::const_reference` is; presumably a reference-to-const T for `container<T>`.

#### Returns

This iterator, for chained operations.

This kind of iterator doesn't really have a *position* in the container (you can think of the position as being permanently at the end, if you like). Assigning a value to the iterator will always append the value to the end of the container.

Definition at line 430 of file `stl_iterator.h`.

The documentation for this class was generated from the following file:

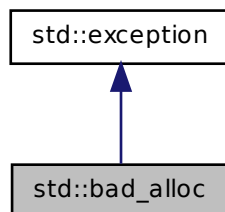
- [stl\\_iterator.h](#)

## 5.370 `std::bad_alloc` Class Reference

Exception possibly thrown by `new`.

`bad_alloc` (or classes derived from it) is used to report allocation errors from the throwing forms of `new`.

Inheritance diagram for `std::bad_alloc`:



### Public Member Functions

- virtual const char \* `what` () const throw ()

#### 5.370.1 Detailed Description

Exception possibly thrown by `new`.

`bad_alloc` (or classes derived from it) is used to report allocation errors from the throwing forms of `new`.

Definition at line 56 of file `new`.

### 5.370.2 Member Function Documentation

#### 5.370.2.1 `virtual const char* std::bad_alloc::what ( ) const throw ()` [`virtual`]

Returns a C-style character string describing the general cause of the current error.

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

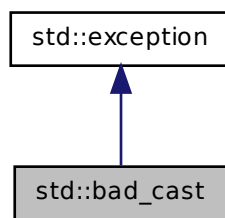
- [new](#)

### 5.371 `std::bad_cast` Class Reference

Thrown during incorrect typecasting.

If you attempt an invalid `dynamic_cast` expression, an instance of this class (or something derived from this class) is thrown.

Inheritance diagram for `std::bad_cast`:



#### Public Member Functions

- `virtual const char * what () const throw ()`

#### 5.371.1 Detailed Description

Thrown during incorrect typecasting.

If you attempt an invalid `dynamic_cast` expression, an instance of this class (or something derived from this class) is thrown.

Definition at line 190 of file `typeinfo`.

### 5.371.2 Member Function Documentation

#### 5.371.2.1 `virtual const char* std::bad_cast::what ( ) const throw ()` [`virtual`]

Returns a C-style character string describing the general cause of the current error.

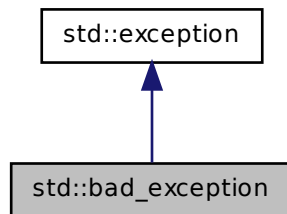
Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [typeinfo](#)

## 5.372 `std::bad_exception` Class Reference

Inheritance diagram for `std::bad_exception`:



### Public Member Functions

- `virtual const char * what () const throw ()`

### 5.372.1 Detailed Description

If an exception is thrown which is not listed in a function's exception specification, one of these may be thrown.

Definition at line 74 of file `exception`.

### 5.372.2 Member Function Documentation

#### 5.372.2.1 `virtual const char* std::bad_exception::what ( ) const throw ()` [`virtual`]

Returns a C-style character string describing the general cause of the current error.

Reimplemented from [std::exception](#).

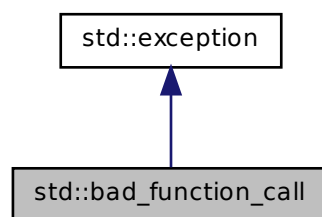
The documentation for this class was generated from the following file:

- [exception](#)

## 5.373 `std::bad_function_call` Class Reference

Exception class thrown when class template function's `operator()` is called with an empty target.

Inheritance diagram for `std::bad_function_call`:



### Public Member Functions

- `virtual const char * what () const throw ()`

### 5.373.1 Detailed Description

Exception class thrown when class template function's `operator()` is called with an empty target.

Definition at line 1476 of file `functional`.

### 5.373.2 Member Function Documentation

#### 5.373.2.1 `virtual const char* std::exception::what ( ) const throw ()` [`virtual`, `inherited`]

Returns a C-style character string describing the general cause of the current error.

Reimplemented in `std::bad_exception`, `std::bad_alloc`, `std::bad_cast`, `std::bad_typeid`, `std::future_error`, `std::logic_error`, `std::runtime_error`, `std::ios_base::failure`, and `std::bad_weak_ptr`.

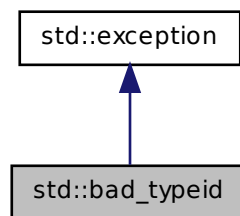
The documentation for this class was generated from the following file:

- `functional`

## 5.374 `std::bad_typeid` Class Reference

Thrown when a NULL pointer in a `typeid` expression is used.

Inheritance diagram for `std::bad_typeid`:



### Public Member Functions

- `virtual const char * what () const throw ()`

### 5.374.1 Detailed Description

Thrown when a NULL pointer in a `typeid` expression is used.

Definition at line 207 of file `typeinfo`.

### 5.374.2 Member Function Documentation

#### 5.374.2.1 `virtual const char* std::bad_typeid::what ( ) const throw ()` [`virtual`]

Returns a C-style character string describing the general cause of the current error.

Reimplemented from [std::exception](#).

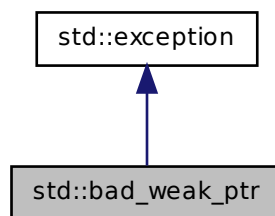
The documentation for this class was generated from the following file:

- [typeinfo](#)

## 5.375 `std::bad_weak_ptr` Class Reference

Exception possibly thrown by [shared\\_ptr](#).

Inheritance diagram for `std::bad_weak_ptr`:



### Public Member Functions

- `virtual char const * what () const throw ()`



### 5.375.1 Detailed Description

Exception possibly thrown by [shared\\_ptr](#).

Definition at line 60 of file `shared_ptr_base.h`.

### 5.375.2 Member Function Documentation

#### 5.375.2.1 `virtual char const* std::bad_weak_ptr::what ( ) const throw ()` [`inline`, `virtual`]

Returns a C-style character string describing the general cause of the current error.

Reimplemented from [std::exception](#).

Definition at line 64 of file `shared_ptr_base.h`.

The documentation for this class was generated from the following file:

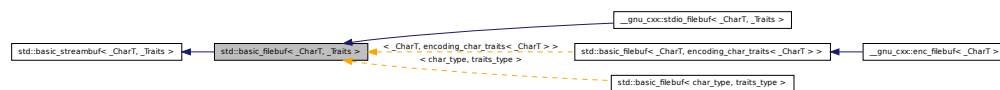
- [shared\\_ptr\\_base.h](#)

## 5.376 `std::basic_filebuf< _CharT, _Traits >` Class Template Reference

The actual work of input and output (for files).

This class associates both its input and output sequence with an external disk file, and maintains a joint file position for both sequences. Many of its semantics are described in terms of similar behavior in the Standard C Library's `FILE` streams.

Inheritance diagram for `std::basic_filebuf< _CharT, _Traits >`:



### Public Types

- typedef `codecvt< char_type, char, __state_type >` `__codecvt_type`
- typedef `__basic_file< char >` `__file_type`
- typedef `basic_filebuf< char_type, traits_type >` `__filebuf_type`
- typedef `traits_type::state_type` `__state_type`
- typedef `basic_streambuf< char_type, traits_type >` `__streambuf_type`

- typedef `_CharT` [char\\_type](#)
- typedef `traits_type::int_type` [int\\_type](#)
- typedef `traits_type::off_type` [off\\_type](#)
- typedef `traits_type::pos_type` [pos\\_type](#)
- typedef `_Traits` [traits\\_type](#)

### Public Member Functions

- [basic\\_filebuf](#) ()
- virtual [~basic\\_filebuf](#) ()
- [\\_\\_filebuf\\_type](#) \* [close](#) ()
- [streamsize](#) [in\\_avail](#) ()
- [bool](#) [is\\_open](#) () const throw ()
- [\\_\\_filebuf\\_type](#) \* [open](#) (const [char](#) \* \_\_s, [ios\\_base::openmode](#) \_\_mode)
- [\\_\\_filebuf\\_type](#) \* [open](#) (const [std::string](#) & \_\_s, [ios\\_base::openmode](#) \_\_mode)
- [int\\_type](#) [sbumpc](#) ()
- [int\\_type](#) [sgetc](#) ()
- [streamsize](#) [sgetn](#) ([char\\_type](#) \* \_\_s, [streamsize](#) \_\_n)
- [int\\_type](#) [snextc](#) ()
- [int\\_type](#) [sputbackc](#) ([char\\_type](#) \_\_c)
- [int\\_type](#) [putc](#) ([char\\_type](#) \_\_c)
- [streamsize](#) [sputn](#) (const [char\\_type](#) \* \_\_s, [streamsize](#) \_\_n)
- [int\\_type](#) [sungetc](#) ()

### Protected Member Functions

- [void](#) [\\_M\\_allocate\\_internal\\_buffer](#) ()
- [bool](#) [\\_M\\_convert\\_to\\_external](#) ([char\\_type](#) \*, [streamsize](#))
- [void](#) [\\_M\\_create\\_pback](#) ()
- [void](#) [\\_M\\_destroy\\_internal\\_buffer](#) () throw ()
- [void](#) [\\_M\\_destroy\\_pback](#) () throw ()
- [int](#) [\\_M\\_get\\_ext\\_pos](#) ([\\_\\_state\\_type](#) & \_\_state)
- [pos\\_type](#) [\\_M\\_seek](#) ([off\\_type](#) \_\_off, [ios\\_base::seekdir](#) \_\_way, [\\_\\_state\\_type](#) \_\_state)
- [void](#) [\\_M\\_set\\_buffer](#) ([streamsize](#) \_\_off)
- [bool](#) [\\_M\\_terminate\\_output](#) ()
- [void](#) [gbump](#) (int \_\_n)
- virtual [void](#) [imbue](#) (const [locale](#) & \_\_loc)
- virtual [int\\_type](#) [overflow](#) ([int\\_type](#) \_\_c=\_Traits::eof())
- virtual [int\\_type](#) [pbackfail](#) ([int\\_type](#) \_\_c=\_Traits::eof())
- [void](#) [pbump](#) (int \_\_n)

- virtual `pos_type seekoff` (`off_type __off`, `ios_base::seekdir __way`, `ios_base::openmode __mode=ios_base::in|ios_base::out`)
  - virtual `pos_type seekpos` (`pos_type __pos`, `ios_base::openmode __mode=ios_base::in|ios_base::out`)
  - virtual `__streambuf_type * setbuf` (`char_type *__s`, `streamsize __n`)
  - void `setg` (`char_type *__gbeg`, `char_type *__gnext`, `char_type *__gend`)
  - void `setp` (`char_type *__pbeg`, `char_type *__pend`)
  - virtual `streamsize showmanyc` ()
  - virtual `int sync` ()
  - virtual `int_type uflow` ()
  - virtual `int_type underflow` ()
  - virtual `streamsize xsgetn` (`char_type *__s`, `streamsize __n`)
  - virtual `streamsize xsputn` (`const char_type *__s`, `streamsize __n`)
- 
- `char_type * eback` () const
  - `char_type * gptr` () const
  - `char_type * egptr` () const
- 
- `char_type * pbase` () const
  - `char_type * pptr` () const
  - `char_type * epptr` () const

### Protected Attributes

- `char_type * _M_buf`
  - `bool _M_buf_allocated`
  - `size_t _M_buf_size`
  - `const __codecvt_type * _M_codecvt`
  - `char * _M_ext_buf`
  - `streamsize _M_ext_buf_size`
  - `char * _M_ext_end`
  - `const char * _M_ext_next`
  - `__file_type _M_file`
  - `__c_lock _M_lock`
  - `ios_base::openmode _M_mode`
  - `bool _M_reading`
  - `__state_type _M_state_beg`
  - `__state_type _M_state_cur`
  - `__state_type _M_state_last`
  - `bool _M_writing`
- 
- `char_type _M_pback`
  - `char_type * _M_pback_cur_save`
  - `char_type * _M_pback_end_save`
  - `bool _M_pback_init`

## Friends

- template<bool \_IsMove, typename \_CharT2 >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_char< \_CharT2 >::\_\_value, \_CharT2 \* >::\_\_type \_\_copy\_move\_a2(istreambuf\_iterator< \_CharT2 >, istreambuf\_iterator< \_CharT2 >, \_CharT2 \*)
  - streamsize \_\_copy\_streambufs\_eof(\_\_streambuf\_type \*, \_\_streambuf\_type \*, bool &)
  - class basic\_ios< char\_type, traits\_type >
  - class basic\_istream< char\_type, traits\_type >
  - class basic\_ostream< char\_type, traits\_type >
  - template<typename \_CharT2 >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_char< \_CharT2 >::\_\_value, istreambuf\_iterator< \_CharT2 > >::\_\_type find(istreambuf\_iterator< \_CharT2 >, istreambuf\_iterator< \_CharT2 >, const \_CharT2 &)
  - template<typename \_CharT2, typename \_Traits2, typename \_Alloc >  
basic\_istream< \_CharT2, \_Traits2 > & getline(basic\_istream< \_CharT2, \_Traits2 > &, basic\_string< \_CharT2, \_Traits2, \_Alloc > &, \_CharT2)
  - class ios\_base
  - class istreambuf\_iterator< char\_type, traits\_type >
  - template<typename \_CharT2, typename \_Traits2 >  
basic\_istream< \_CharT2, \_Traits2 > & operator>>(basic\_istream< \_CharT2, \_Traits2 > &, \_CharT2 \*)
  - template<typename \_CharT2, typename \_Traits2, typename \_Alloc >  
basic\_istream< \_CharT2, \_Traits2 > & operator>>(basic\_istream< \_CharT2, \_Traits2 > &, basic\_string< \_CharT2, \_Traits2, \_Alloc > &)
  - class ostreambuf\_iterator< char\_type, traits\_type >
- 
- char\_type \* \_M\_in\_beg
  - char\_type \* \_M\_in\_cur
  - char\_type \* \_M\_in\_end
  - char\_type \* \_M\_out\_beg
  - char\_type \* \_M\_out\_cur
  - char\_type \* \_M\_out\_end
  - locale \_M\_buf\_locale
  - locale pubimbue(const locale &\_\_loc)
  - locale getloc() const
  - \_\_streambuf\_type \* pubsetbuf(char\_type \*\_\_s, streamsize \_\_n)
  - pos\_type pubseekoff(off\_type \_\_off, ios\_base::seekdir \_\_way, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
  - pos\_type pubseekpos(pos\_type \_\_sp, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
  - int pubsync()

### 5.376.1 Detailed Description

**template<typename \_CharT, typename \_Traits> class std::basic\_filebuf< \_CharT, \_Traits >**

The actual work of input and output (for files).

This class associates both its input and output sequence with an external disk file, and maintains a joint file position for both sequences. Many of its semantics are described in terms of similar behavior in the Standard C Library's `FILE` streams.

Definition at line 69 of file `fstream`.

### 5.376.2 Member Typedef Documentation

**5.376.2.1   template<typename \_CharT, typename \_Traits> typedef  
              basic\_streambuf<char\_type, traits\_type> std::basic\_filebuf<  
              \_CharT, \_Traits >::\_\_streambuf\_type**

This is a non-standard type.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 79 of file `fstream`.

**5.376.2.2   template<typename \_CharT, typename \_Traits> typedef \_CharT  
              std::basic\_filebuf< \_CharT, \_Traits >::char\_type**

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Reimplemented in [\\_\\_gnu\\_cxx::stdio\\_filebuf< \\_CharT, \\_Traits >](#).

Definition at line 73 of file `fstream`.

**5.376.2.3   template<typename \_CharT, typename \_Traits> typedef  
              traits\_type::int\_type std::basic\_filebuf< \_CharT, \_Traits >::int\_type**

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Reimplemented in [\\_\\_gnu\\_cxx::stdio\\_filebuf< \\_CharT, \\_Traits >](#).

Definition at line 75 of file `fstream`.

**5.376.2.4 `template<typename _CharT, typename _Traits> typedef traits_type::off_type std::basic_filebuf< _CharT, _Traits >::off_type`**

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Reimplemented in [\\_\\_gnu\\_cxx::stdio\\_filebuf< \\_CharT, \\_Traits >](#).

Definition at line 77 of file `fstream`.

**5.376.2.5 `template<typename _CharT, typename _Traits> typedef traits_type::pos_type std::basic_filebuf< _CharT, _Traits >::pos_type`**

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Reimplemented in [\\_\\_gnu\\_cxx::enc\\_filebuf< \\_CharT >](#), and [\\_\\_gnu\\_cxx::stdio\\_filebuf< \\_CharT, \\_Traits >](#).

Definition at line 76 of file `fstream`.

**5.376.2.6 `template<typename _CharT, typename _Traits> typedef _Traits std::basic_filebuf< _CharT, _Traits >::traits_type`**

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Reimplemented in [\\_\\_gnu\\_cxx::enc\\_filebuf< \\_CharT >](#), and [\\_\\_gnu\\_cxx::stdio\\_filebuf< \\_CharT, \\_Traits >](#).

Definition at line 74 of file `fstream`.

### 5.376.3 Constructor & Destructor Documentation

#### 5.376.3.1 `template<typename _CharT, typename _Traits > std::basic_filebuf< _CharT, _Traits >::basic_filebuf ( )`

Does not open any files.

The default constructor initializes the parent class using its own default ctor.

Definition at line 81 of file fstream.tcc.

References std::basic\_streambuf< \_CharT, \_Traits >::\_M\_buf\_locale.

#### 5.376.3.2 `template<typename _CharT, typename _Traits> virtual std::basic_filebuf< _CharT, _Traits >::~~basic_filebuf ( ) [inline, virtual]`

The destructor closes the file first.

Definition at line 216 of file fstream.

### 5.376.4 Member Function Documentation

#### 5.376.4.1 `template<typename _CharT, typename _Traits> void std::basic_filebuf< _CharT, _Traits >::_M_create_pback ( ) [inline, protected]`

Initializes pback buffers, and moves normal buffers to safety. Assumptions: \_M\_in\_cur has already been moved back

Definition at line 174 of file fstream.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::pbackfail().

#### 5.376.4.2 `template<typename _CharT, typename _Traits> void std::basic_filebuf< _CharT, _Traits >::_M_destroy_pback ( ) throw () [inline, protected]`

Deactivates pback buffer contents, and restores normal buffer. Assumptions: The pback buffer has only moved forward.

Definition at line 191 of file fstream.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::overflow(), std::basic\_filebuf< \_CharT, \_Traits >::seekoff(), std::basic\_filebuf< \_CharT, \_Traits >::seekpos(),

std::basic\_filebuf< \_CharT, \_Traits >::underflow(), and std::basic\_filebuf< \_CharT, \_Traits >::xsgetn().

#### 5.376.4.3 template<typename \_CharT, typename \_Traits> void std::basic\_filebuf< \_CharT, \_Traits >::\_M\_set\_buffer ( streamsize \_\_off ) [inline, protected]

This function sets the pointers of the internal buffer, both get and put areas. Typically: \_\_off == [egptr\(\)](#) - [eback\(\)](#) upon underflow/uflow (**read** mode); \_\_off == 0 upon overflow (**write** mode); \_\_off == -1 upon open, setbuf, seekoff/pos (**uncommitted** mode).

NB: [epptr\(\)](#) - [pbase\(\)](#) == \_M\_buf\_size - 1, since \_M\_buf\_size reflects the actual allocated memory and the last cell is reserved for the overflow char of a full put area.

Definition at line 392 of file fstream.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::imbue(), std::basic\_filebuf< \_CharT, \_Traits >::open(), std::basic\_filebuf< \_CharT, \_Traits >::overflow(), std::basic\_filebuf< \_CharT, \_Traits >::pbackfail(), [\\_\\_gnu\\_cxx::stdio\\_filebuf< \\_CharT, \\_Traits >::stdio\\_filebuf\(\)](#), std::basic\_filebuf< \_CharT, \_Traits >::underflow(), std::basic\_filebuf< \_CharT, \_Traits >::xsgetn(), and std::basic\_filebuf< \_CharT, \_Traits >::xspn().

#### 5.376.4.4 template<typename \_CharT, typename \_Traits > basic\_filebuf< \_CharT, \_Traits >::\_filebuf\_type \* std::basic\_filebuf< \_CharT, \_Traits >::close ( )

Closes the currently associated file.

#### Returns

this on success, NULL on failure

If no file is currently open, this function immediately fails.

If a *put buffer area* exists, [overflow\(eof\)](#) is called to flush all the characters. The file is then closed.

If any operations fail, this function also fails.

Definition at line 130 of file fstream.tcc.

References std::basic\_filebuf< \_CharT, \_Traits >::is\_open().

Referenced by std::basic\_fstream< \_CharT, \_Traits >::close(), std::basic\_ofstream< \_CharT, \_Traits >::close(), std::basic\_ifstream< \_CharT, \_Traits >::close(),



std::basic\_filebuf< \_CharT, \_Traits >::open(), and std::basic\_filebuf< char\_type, traits\_type >::~~basic\_filebuf().

#### 5.376.4.5 template<typename \_CharT, typename \_Traits> char\_type\* std::basic\_streambuf< \_CharT, \_Traits >::eback ( ) const [inline, protected, inherited]

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- [eback\(\)](#) returns the beginning pointer for the input sequence
- [gptr\(\)](#) returns the next pointer for the input sequence
- [egptr\(\)](#) returns the end pointer for the input sequence

Definition at line 462 of file streambuf.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::\_M\_destroy\_pback(), std::basic\_filebuf< \_CharT, \_Traits >::imbue(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::overflow(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::pbackfail(), std::basic\_filebuf< \_CharT, \_Traits >::pbackfail(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos(), std::basic\_streambuf< char\_type, traits\_type >::sputbackc(), std::basic\_streambuf< char\_type, traits\_type >::sungetc(), std::basic\_filebuf< \_CharT, \_Traits >::underflow(), and std::basic\_filebuf< \_CharT, \_Traits >::xsgetn().

#### 5.376.4.6 template<typename \_CharT, typename \_Traits> char\_type\* std::basic\_streambuf< \_CharT, \_Traits >::egptr ( ) const [inline, protected, inherited]

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- [eback\(\)](#) returns the beginning pointer for the input sequence
- [gptr\(\)](#) returns the next pointer for the input sequence
- [egptr\(\)](#) returns the end pointer for the input sequence

Definition at line 468 of file streambuf.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::\_M\_create\_pback(), std::basic\_streambuf< char\_type, traits\_type >::in\_avail(), std::basic\_streambuf< char\_type, traits\_type >::sbumpc(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos(), std::basic\_streambuf< char\_type, traits\_type >::sgetc(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::showmanyc(), std::basic\_filebuf< \_CharT, \_Traits >::showmanyc(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::str(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::underflow(), std::basic\_filebuf< \_CharT, \_Traits >::underflow(), std::basic\_streambuf< \_CharT, \_Traits >::xsgetn(), and std::basic\_filebuf< \_CharT, \_Traits >::xsgetn().

**5.376.4.7** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::epptr ( ) const  
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- [pbase\(\)](#) returns the beginning pointer for the output sequence
- [pptr\(\)](#) returns the next pointer for the output sequence
- [epptr\(\)](#) returns the end pointer for the output sequence

Definition at line 515 of file streambuf.

Referenced by std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::overflow(), std::basic\_streambuf< char\_type, traits\_type >::sputc(), std::basic\_streambuf< \_CharT, \_Traits >::xsputn(), and std::basic\_filebuf< \_CharT, \_Traits >::xsputn().

**5.376.4.8** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf< _CharT, _Traits >::gbump ( int __n )  
[inline, protected, inherited]`

Moving the read position.

#### Parameters

- n* The delta by which to move.

This just advances the read position without returning any data.

Definition at line 478 of file `streambuf`.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_streambuf< char_type, traits_type >::sbumpc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_streambuf< char_type, traits_type >::sputbackc()`, `std::basic_streambuf< char_type, traits_type >::sungetc()`, `std::basic_streambuf< char_type, traits_type >::uflow()`, `std::basic_streambuf< _CharT, _Traits >::xsgetn()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

**5.376.4.9** `template<typename _CharT, typename _Traits> locale  
std::basic_streambuf< _CharT, _Traits >::getloc ( ) const  
[inline, inherited]`

Locale access.

#### Returns

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 224 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::pubimbue()`.

**5.376.4.10** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::gptr ( ) const  
[inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 465 of file streambuf.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::\_M\_create\_pback(), std::basic\_filebuf< char\_type, traits\_type >::\_M\_destroy\_pback(), std::basic\_filebuf< \_CharT, \_Traits >::imbue(), std::basic\_streambuf< char\_type, traits\_type >::in\_avail(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::overflow(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::pbackfail(), std::basic\_filebuf< \_CharT, \_Traits >::pbackfail(), std::basic\_streambuf< char\_type, traits\_type >::sbumpc(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos(), std::basic\_streambuf< char\_type, traits\_type >::sgetc(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::showmanyc(), std::basic\_filebuf< \_CharT, \_Traits >::showmanyc(), std::basic\_streambuf< char\_type, traits\_type >::sputbackc(), std::basic\_streambuf< char\_type, traits\_type >::sungetc(), std::basic\_streambuf< char\_type, traits\_type >::uflow(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::underflow(), std::basic\_filebuf< \_CharT, \_Traits >::underflow(), std::basic\_streambuf< \_CharT, \_Traits >::xsgetn(), and std::basic\_filebuf< \_CharT, \_Traits >::xsgetn().

**5.376.4.11** `template<typename _CharT, typename _Traits > void  
std::basic_filebuf< _CharT, _Traits >::imbue ( const locale & )  
[protected, virtual]`

Changes translations.

#### Parameters

*loc* A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

#### Note

Base class version does nothing.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 914 of file fstream.tcc.

References std::basic\_filebuf< \_CharT, \_Traits >::\_M\_ext\_buf, std::basic\_filebuf< \_CharT, \_Traits >::\_M\_ext\_next, std::basic\_filebuf< \_CharT, \_Traits >::\_M\_mode, std::basic\_filebuf< \_CharT, \_Traits >::\_M\_reading, std::basic\_filebuf< \_CharT, \_Traits >::\_M\_set\_buffer(), std::ios\_base::cur, std::basic\_streambuf< \_CharT, \_Traits >::eback(), std::basic\_streambuf< \_CharT, \_Traits >::gptr(), std::basic\_filebuf< \_CharT, \_Traits >::is\_open(), and std::basic\_filebuf< \_CharT, \_Traits >::seekoff().

**5.376.4.12** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf< _CharT, _Traits >::in_avail ( )  
[inline, inherited]`

Looking ahead into the stream.

#### Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

Definition at line 264 of file streambuf.

**5.376.4.13** `template<typename _CharT, typename _Traits> bool  
std::basic_filebuf< _CharT, _Traits >::is_open ( ) const throw ()  
[inline]`

Returns true if the external file is open.

Definition at line 224 of file fstream.

Referenced by `std::basic_filebuf< _CharT, _Traits >::close()`, `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_fstream< _CharT, _Traits >::is_open()`, `std::basic_ofstream< _CharT, _Traits >::is_open()`, `std::basic_ifstream< _CharT, _Traits >::is_open()`, `std::basic_filebuf< _CharT, _Traits >::open()`, `std::basic_filebuf< _CharT, _Traits >::seekoff()`, `std::basic_filebuf< _CharT, _Traits >::seekpos()`, `std::basic_filebuf< _CharT, _Traits >::setbuf()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, and `__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf()`.

**5.376.4.14** `template<typename _CharT, typename _Traits > basic_filebuf<  
_CharT, _Traits >::_filebuf_type * std::basic_filebuf< _CharT,  
_Traits >::open ( const char * __s, ios_base::openmode __mode )`

Opens an external file.

#### Parameters

*s* The name of the file.

*mode* The open mode flags.

**Returns**

`this` on success, NULL on failure

If a file is already open, this function immediately fails. Otherwise it tries to open the file named *s* using the flags given in *mode*.

Table 92, adapted here, gives the relation between openmode combinations and the equivalent fopen() flags. (NB: lines app, in|out|app, in|app, binary|app, binary|in|out|app, and binary|in|app per DR 596) +-----+  
+-----+ | ios\_base Flag combination | | binary in out trunc app  
| +-----+ | + w | | + a | | + a | | + w | | +  
r | | + r+ | | + + w+ | | + + a+ | | + + a+ | +-----+  
+-----+ | + + wb | | + + ab | | + + ab | | + + wb | | + + rb | | + + r+b | | + + +  
w+b | | + + + a+b | | + + + a+b | +-----+

Definition at line 96 of file fstream.tcc.

References std::basic\_filebuf< \_CharT, \_Traits >::\_M\_mode, std::basic\_filebuf< \_CharT, \_Traits >::\_M\_reading, std::basic\_filebuf< \_CharT, \_Traits >::\_M\_set\_buffer(), std::ios\_base::ate, std::basic\_filebuf< \_CharT, \_Traits >::close(), std::ios\_base::end, std::basic\_filebuf< \_CharT, \_Traits >::is\_open(), and std::basic\_filebuf< \_CharT, \_Traits >::seekoff().

Referenced by std::basic\_fstream< \_CharT, \_Traits >::open(), std::basic\_ofstream< \_CharT, \_Traits >::open(), and std::basic\_ifstream< \_CharT, \_Traits >::open().

**5.376.4.15** `template<typename _CharT, typename _Traits> __filebuf_type*  
std::basic_filebuf< _CharT, _Traits >::open ( const std::string &  
__s, ios_base::openmode __mode ) [inline]`

Opens an external file.

**Parameters**

*s* The name of the file.  
*mode* The open mode flags.

**Returns**

`this` on success, NULL on failure

Definition at line 277 of file fstream.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::open().

**5.376.4.16** `template<typename _CharT, typename _Traits > basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::overflow ( int_type = _Traits::eof() ) [protected, virtual]`

Consumes data from the buffer; writes to the controlled sequence.

#### Parameters

*c* An additional character to consume.

#### Returns

`eof()` to indicate failure, something else (usually *c*, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if *c* is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

#### Note

Base class version does nothing, returns `eof()`.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 424 of file `fstream.tcc`.

References `std::basic_filebuf< _CharT, _Traits >::_M_buf_size`, `std::basic_filebuf< _CharT, _Traits >::_M_destroy_pback()`, `std::basic_filebuf< _CharT, _Traits >::_M_mode`, `std::basic_filebuf< _CharT, _Traits >::_M_reading`, `std::basic_filebuf< _CharT, _Traits >::_M_set_buffer()`, `std::ios_base::cur`, `std::ios_base::out`, `std::basic_streambuf< _CharT, _Traits >::pbase()`, `std::basic_streambuf< _CharT, _Traits >::pbump()`, and `std::basic_streambuf< _CharT, _Traits >::pptr()`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_filebuf< _CharT, _Traits >::sync()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

**5.376.4.17** `template<typename _CharT, typename _Traits> basic_filebuf<_CharT, _Traits>::int_type std::basic_filebuf<_CharT, _Traits>::pbackfail( int_type = _Traits::eof() ) [protected, virtual]`

Tries to back up the input sequence.

#### Parameters

*c* The character to be inserted back into the sequence.

#### Returns

eof() on failure, *some other value* on success

#### Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

#### Note

Base class version does nothing, returns eof().

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 365 of file fstream.tcc.

References `std::basic_filebuf< _CharT, _Traits >::_M_create_pback()`, `std::basic_filebuf< _CharT, _Traits >::_M_mode`, `std::basic_filebuf< _CharT, _Traits >::_M_pback_init`, `std::basic_filebuf< _CharT, _Traits >::_M_reading`, `std::basic_filebuf< _CharT, _Traits >::_M_set_buffer()`, `std::ios_base::cur`, `std::basic_streambuf< _CharT, _Traits >::eback()`, `std::basic_streambuf< _CharT, _Traits >::gbump()`, `std::basic_streambuf< _CharT, _Traits >::gptr()`, `std::ios_base::in`, `std::basic_filebuf< _CharT, _Traits >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::seekoff()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

**5.376.4.18** `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::pbase( ) const [inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.



- [pbase\(\)](#) returns the beginning pointer for the output sequence
- [pptr\(\)](#) returns the next pointer for the output sequence
- [epptr\(\)](#) returns the end pointer for the output sequence

Definition at line 509 of file streambuf.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_filebuf<_CharT, _Traits>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::str()`, `std::basic_filebuf<_CharT, _Traits>::sync()`, and `std::basic_filebuf<_CharT, _Traits>::xsputn()`.

**5.376.4.19** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf<_CharT, _Traits>::pbump ( int __n )  
[inline, protected, inherited]`

Moving the write position.

#### Parameters

*n* The delta by which to move.

This just advances the write position without returning any data.

Definition at line 525 of file streambuf.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, `std::basic_streambuf<char_type, traits_type>::sputc()`, and `std::basic_streambuf<_CharT, _Traits>::xsputn()`.

**5.376.4.20** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::pptr ( ) const  
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- [pbase\(\)](#) returns the beginning pointer for the output sequence
- [pptr\(\)](#) returns the next pointer for the output sequence
- [epptr\(\)](#) returns the end pointer for the output sequence

Definition at line 512 of file streambuf.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_filebuf< _CharT, _Traits >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_streambuf< char_type, traits_type >::sputc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::str()`, `std::basic_filebuf< _CharT, _Traits >::sync()`, `std::basic_streambuf< _CharT, _Traits >::xsputn()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

**5.376.4.21** `template<typename _CharT, typename _Traits> locale  
std::basic_streambuf< _CharT, _Traits >::pubimbue ( const locale  
& __loc ) [inline, inherited]`

Entry point for [imbue\(\)](#).

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 207 of file streambuf.

**5.376.4.22** `template<typename _CharT, typename _Traits> pos_type  
std::basic_streambuf< _CharT, _Traits >::pubseekoff ( off_type  
__off, ios_base::seekdir __way, ios_base::openmode __mode =  
ios_base::in | ios_base::out ) [inline, inherited]`

Entry point for [imbue\(\)](#).

#### Parameters

*loc* The new locale.

**Returns**

The previous locale.

Calls the derived imbue(loc).

Definition at line 241 of file streambuf.

**5.376.4.23** `template<typename _CharT, typename _Traits> pos_type  
std::basic_streambuf<_CharT, _Traits>::pubseekpos ( pos_type  
__sp, ios_base::openmode __mode = ios_base::in | ios_base::out )  
[inline, inherited]`

Entry point for [imbue\(\)](#).

**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

Calls the derived imbue(loc).

Definition at line 246 of file streambuf.

**5.376.4.24** `template<typename _CharT, typename _Traits>  
__streambuf_type* std::basic_streambuf<_CharT, _Traits  
>::pubsetbuf ( char_type * __s, streamsize __n ) [inline,  
inherited]`

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 237 of file streambuf.

**5.376.4.25** `template<typename _CharT, typename _Traits> int  
std::basic_streambuf<_CharT, _Traits>::pubsync ( )  
[inline, inherited]`

Entry point for [imbue\(\)](#).

**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 251 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::sync()`.

**5.376.4.26** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf< _CharT, _Traits >::sbumpc( ) [inline,  
inherited]`

Getting the next character.

**Returns**

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 296 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::istreambuf_iterator< _CharT, _Traits >::operator++()`, and `std::basic_streambuf< char_type, traits_type >::snextc()`.

**5.376.4.27** `template<typename _CharT, typename _Traits > basic_filebuf<  
_CharT, _Traits >::pos_type std::basic_filebuf< _CharT, _Traits  
>::seekoff( off_type, ios_base::seekdir, ios_base::openmode =  
ios_base::in | ios_base::out ) [protected, virtual]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 717 of file `fstream.tcc`.

References `std::basic_filebuf< _CharT, _Traits >::_M_destroy_pback()`, `std::basic_filebuf< _CharT, _Traits >::_M_reading`, `std::ios_base::cur`, `std::basic_filebuf< _CharT, _Traits >::is_open()`, `std::basic_streambuf< _CharT, _Traits >::pbase()`, and `std::basic_streambuf< _CharT, _Traits >::pptr()`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::open()`, and `std::basic_filebuf< _CharT, _Traits >::pbackfail()`.

**5.376.4.28** `template<typename _CharT, typename _Traits > basic_filebuf< _CharT, _Traits >::pos_type std::basic_filebuf< _CharT, _Traits >::seekpos ( pos_type, ios_base::openmode = ios_base::in | ios_base::out ) [protected, virtual]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

#### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 777 of file `fstream.tcc`.

References `std::basic_filebuf< _CharT, _Traits >::_M_destroy_pback()`, `std::ios_base::beg`, and `std::basic_filebuf< _CharT, _Traits >::is_open()`.

**5.376.4.29** `template<typename _CharT, typename _Traits > basic_filebuf< _CharT, _Traits >::_streambuf_type * std::basic_filebuf< _CharT, _Traits >::setbuf ( char_type * __s, streamsize __n ) [protected, virtual]`

Manipulates the buffer.

#### Parameters

*s* Pointer to a buffer area.

*n* Size of *s*.

**Returns**

`this`

If no file has been opened, and both *s* and *n* are zero, then the stream becomes unbuffered. Otherwise, *s* is used as a buffer; see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 688 of file `fstream.tcc`.

References `std::basic_filebuf< _CharT, _Traits >::_M_buf`, `std::basic_filebuf< _CharT, _Traits >::_M_buf_size`, and `std::basic_filebuf< _CharT, _Traits >::is_open()`.

**5.376.4.30** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf< _CharT, _Traits >::setg ( char_type *  
__gbeg, char_type * __gnext, char_type * __gend ) [inline,  
protected, inherited]`

Setting the three read area pointers.

**Parameters**

*gbeg* A pointer.

*gnext* A pointer.

*gend* A pointer.

**Postcondition**

*gbeg* == `eback()`, *gnext* == `gptr()`, and *gend* == `egptr()`

Definition at line 489 of file `streambuf`.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_create_pback()`, `std::basic_filebuf< char_type, traits_type >::_M_destroy_pback()`, and `std::basic_filebuf< char_type, traits_type >::_M_set_buffer()`.

**5.376.4.31** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf< _CharT, _Traits >::setp ( char_type  
* __pbeg, char_type * __pend ) [inline, protected,  
inherited]`

Setting the three write area pointers.

**Parameters**

*pbeg* A pointer.

*pend* A pointer.

**Postcondition**

*pbeg* == `pbase()`, *pbeg* == `pptr()`, and *pend* == `epptr()`

Definition at line 535 of file streambuf.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_set_buffer()`.

**5.376.4.32** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf< _CharT, _Traits >::sgetc ( ) [inline,  
inherited]`

Getting the next character.

**Returns**

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 318 of file streambuf.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_streambuf< char_type, traits_type >::snextc()`.

**5.376.4.33** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf< _CharT, _Traits >::sgetn ( char_type * __s,  
streamsize __n ) [inline, inherited]`

Entry point for `xsgetn`.

**Parameters**

*s* A buffer area.

*n* A count.

Returns `xsgetn(s,n)`. The effect is to fill `s[0]` through `s[n-1]` with characters from the input sequence, if possible.

Definition at line 337 of file `streambuf`.

**5.376.4.34** `template<typename _CharT, typename _Traits > streamsize  
std::basic_filebuf< _CharT, _Traits >::showmanyc ( )  
[protected, virtual]`

Investigating the data available.

#### Returns

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.*  
[27.5.2.4.3]/1

#### Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 180 of file `fstream.tcc`.

References `std::basic_filebuf< _CharT, _Traits >::_M_mode`, `std::ios_base::binary`, `std::basic_streambuf< _CharT, _Traits >::egptr()`, `std::basic_streambuf< _CharT, _Traits >::gptr()`, `std::ios_base::in`, and `std::basic_filebuf< _CharT, _Traits >::is_open()`.

**5.376.4.35** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf< _CharT, _Traits >::snextc ( ) [inline,  
inherited]`

Getting the next character.



**Returns**

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 278 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

**5.376.4.36** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf< _CharT, _Traits >::sputbackc ( char_type  
__c ) [inline, inherited]`

Pushing characters back into the input stream.

**Parameters**

*c* The character to push back.

**Returns**

The previous character, if possible.

Similar to `sungetc()`, but *c* is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be *c*.

Definition at line 352 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::putback()`.

**5.376.4.37** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf< _CharT, _Traits >::sputc ( char_type __c )  
[inline, inherited]`

Entry point for all single-character output functions.

**Parameters**

*c* A character to output.

**Returns**

*c*, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores *c* in that position, increments the position, and returns `traits::to_int_type(c)`. If a write position is not available, returns `overflow(c)`.

Definition at line 404 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, and `std::ostreambuf_iterator<_CharT, _Traits>::operator=()`.

**5.376.4.38** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf<_CharT, _Traits>::sputn ( const char_type  
* __s, streamsize __n ) [inline, inherited]`

Entry point for all single-character output functions.

#### Parameters

*s* A buffer read area.

*n* A count.

One of two public output functions.

Returns `xspn(s,n)`. The effect is to write *s*[0] through *s*[*n*-1] to the output sequence, if possible.

Definition at line 430 of file `streambuf`.

**5.376.4.39** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf<_CharT, _Traits>::sungetc ( ) [inline,  
inherited]`

Moving backwards in the input stream.

#### Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbackfail()`. The effect is to *unget* the last character *gotten*.

Definition at line 377 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::unget()`.

**5.376.4.40** `template<typename _CharT, typename _Traits > int  
std::basic_filebuf< _CharT, _Traits >::sync ( ) [protected,  
virtual]`

Synchronizes the buffer arrays with the controlled sequences.

#### Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

#### Note

Base class version does nothing, returns zero.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 897 of file fstream.tcc.

References [std::basic\\_filebuf< \\_CharT, \\_Traits >::overflow\(\)](#), [std::basic\\_streambuf< \\_CharT, \\_Traits >::pbase\(\)](#), and [std::basic\\_streambuf< \\_CharT, \\_Traits >::pptr\(\)](#).

**5.376.4.41** `template<typename _CharT, typename _Traits> virtual int_type  
std::basic_streambuf< _CharT, _Traits >::uflow ( ) [inline,  
protected, virtual, inherited]`

Fetches more data from the controlled sequence.

#### Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as [underflow\(\)](#), and in fact is required to call that function. It also returns the new character, like [underflow\(\)](#) does. However, this function also moves the read position forward by one.

Reimplemented in [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf< \\_CharT, \\_Traits >](#).

Definition at line 680 of file streambuf.

Referenced by [std::basic\\_streambuf< char\\_type, traits\\_type >::sbumpc\(\)](#), and [std::basic\\_streambuf< \\_CharT, \\_Traits >::xsgetn\(\)](#).

**5.376.4.42** `template<typename _CharT, typename _Traits > basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::underflow ( ) [protected, virtual]`

Fetches more data from the controlled sequence.

### Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25.html>

### Note

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 206 of file `fstream.tcc`.

References `std::basic_filebuf< _CharT, _Traits >::_M_buf_size`, `std::basic_filebuf< _CharT, _Traits >::_M_destroy_pback()`, `std::basic_filebuf< _CharT, _Traits >::_M_ext_buf`, `std::basic_filebuf< _CharT, _Traits >::_M_ext_buf_size`, `std::basic_filebuf< _CharT, _Traits >::_M_ext_next`, `std::basic_filebuf< _CharT, _Traits >::_M_mode`, `std::basic_filebuf< _CharT, _Traits >::_M_reading`, `std::basic_filebuf< _CharT, _Traits >::_M_set_buffer()`, `std::basic_streambuf< _CharT, _Traits >::eback()`, `std::basic_streambuf< _CharT, _Traits >::egptr()`, `std::basic_streambuf< _CharT, _Traits >::gptr()`, `std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::in()`, `std::ios_base::in`, `std::min()`, and `std::basic_filebuf< _CharT, _Traits >::overflow()`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::pbackfail()`.

**5.376.4.43** `template<typename _CharT, typename _Traits > streamsize std::basic_filebuf< _CharT, _Traits >::xsgetn ( char_type * __s, streamsize __n ) [protected, virtual]`

Multiple character extraction.

#### Parameters

- s* A buffer area.
- n* Maximum number of characters to assign.

#### Returns

The number of characters assigned.

Fills `s[0]` through `s[n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either *n* characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 551 of file `fstream.tcc`.

References `std::basic_filebuf< _CharT, _Traits >::_M_buf_size`, `std::basic_filebuf< _CharT, _Traits >::_M_destroy_pback()`, `std::basic_filebuf< _CharT, _Traits >::_M_mode`, `std::basic_filebuf< _CharT, _Traits >::_M_pback_init`, `std::basic_filebuf< _CharT, _Traits >::_M_reading`, `std::basic_filebuf< _CharT, _Traits >::_M_set_buffer()`, `std::basic_streambuf< _CharT, _Traits >::eback()`, `std::basic_streambuf< _CharT, _Traits >::egptr()`, `std::basic_streambuf< _CharT, _Traits >::gbump()`, `std::basic_streambuf< _CharT, _Traits >::gptr()`, `std::ios_base::in`, `std::basic_filebuf< _CharT, _Traits >::overflow()`, and `std::basic_streambuf< char_type, traits_type >::xsgetn()`.

**5.376.4.44** `template<typename _CharT, typename _Traits> streamsize  
std::basic_filebuf< _CharT, _Traits >::xspn( const char_type *  
__s, streamsize __n ) [protected, virtual]`

Multiple character insertion.

#### Parameters

- s* A buffer area.
- n* Maximum number of characters to write.

#### Returns

The number of characters written.

Writes `s[0]` through `s[n-1]` to the output sequence, as if by `sputc()`. Stops when either `n` characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 641 of file `fstream.tcc`.

References `std::basic_filebuf<_CharT, _Traits>::_M_buf_size`, `std::basic_filebuf<_CharT, _Traits>::_M_mode`, `std::basic_filebuf<_CharT, _Traits>::_M_reading`, `std::basic_filebuf<_CharT, _Traits>::_M_set_buffer()`, `std::basic_streambuf<_CharT, _Traits>::epptr()`, `std::min()`, `std::ios_base::out`, `std::basic_streambuf<_CharT, _Traits>::pbase()`, `std::basic_streambuf<_CharT, _Traits>::pptr()`, and `std::basic_streambuf<char_type, traits_type>::xsputn()`.

### 5.376.5 Member Data Documentation

#### 5.376.5.1 `template<typename _CharT, typename _Traits> char_type* std::basic_filebuf<_CharT, _Traits>::_M_buf` [protected]

Pointer to the beginning of internal buffer.

Definition at line 111 of file `fstream`.

Referenced by `std::basic_filebuf<char_type, traits_type>::_M_destroy_pback()`, `std::basic_filebuf<char_type, traits_type>::_M_set_buffer()`, and `std::basic_filebuf<_CharT, _Traits>::setbuf()`.

#### 5.376.5.2 `template<typename _CharT, typename _Traits> locale std::basic_streambuf<_CharT, _Traits>::_M_buf_locale` [protected, inherited]

Current locale setting.

Definition at line 190 of file `streambuf`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::basic_filebuf()`, `std::basic_streambuf<char_type, traits_type>::getloc()`, and `std::basic_streambuf<char_type, traits_type>::pubimbue()`.

**5.376.5.3 template<typename \_CharT, typename \_Traits> size\_t  
std::basic\_filebuf< \_CharT, \_Traits >::\_M\_buf\_size [protected]**

Actual size of internal buffer. This number is equal to the size of the put area + 1 position, reserved for the overflow char of a full area.

Definition at line 118 of file fstream.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::\_M\_set\_buffer(), std::basic\_filebuf< \_CharT, \_Traits >::overflow(), std::basic\_filebuf< \_CharT, \_Traits >::setbuf(), \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >::stdio\_filebuf(), std::basic\_filebuf< \_CharT, \_Traits >::underflow(), std::basic\_filebuf< \_CharT, \_Traits >::xsgetn(), and std::basic\_filebuf< \_CharT, \_Traits >::xsputn().

**5.376.5.4 template<typename \_CharT, typename \_Traits> char\*  
std::basic\_filebuf< \_CharT, \_Traits >::\_M\_ext\_buf [protected]**

Buffer for external characters. Used for input when codecvt::always\_noconv() == false. When valid, this corresponds to [eback\(\)](#).

Definition at line 153 of file fstream.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::imbue(), and std::basic\_filebuf< \_CharT, \_Traits >::underflow().

**5.376.5.5 template<typename \_CharT, typename \_Traits> streamsize  
std::basic\_filebuf< \_CharT, \_Traits >::\_M\_ext\_buf\_size  
[protected]**

Size of buffer held by \_M\_ext\_buf.

Definition at line 158 of file fstream.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::underflow().

**5.376.5.6 template<typename \_CharT, typename \_Traits> const  
char\* std::basic\_filebuf< \_CharT, \_Traits >::\_M\_ext\_next  
[protected]**

Pointers into the buffer held by \_M\_ext\_buf that delimit a subsequence of bytes that have been read but not yet converted. When valid, \_M\_ext\_next corresponds to [egptr\(\)](#).

Definition at line 165 of file fstream.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::imbue(), and std::basic\_filebuf< \_CharT, \_Traits >::underflow().

**5.376.5.7** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_in_beg  
[protected, inherited]`

This is based on `_IO_FILE`, just reordered to be more consistent, and is intended to be the most minimal abstraction for an internal buffer.

- `get == input == read`
- `put == output == write`

Definition at line 182 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::eback()`, and `std::basic_streambuf< char_type, traits_type >::setg()`.

**5.376.5.8** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_in_cur  
[protected, inherited]`

Entry point for `imbue()`.

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 183 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::gbump()`, `std::basic_streambuf< char_type, traits_type >::gptr()`, and `std::basic_streambuf< char_type, traits_type >::setg()`.

**5.376.5.9** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_in_end  
[protected, inherited]`

Entry point for `imbue()`.



**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

Calls the derived imbue(loc).

Definition at line 184 of file streambuf.

Referenced by std::basic\_streambuf< char\_type, traits\_type >::egptr(), and std::basic\_streambuf< char\_type, traits\_type >::setg().

**5.376.5.10** `template<typename _CharT, typename _Traits>  
ios_base::openmode std::basic_filebuf< _CharT, _Traits  
>::_M_mode [protected]`

Place to stash in || out || in | out settings for current filebuf.

Definition at line 96 of file fstream.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::\_M\_set\_buffer(), std::basic\_filebuf< \_CharT, \_Traits >::imbue(), std::basic\_filebuf< \_CharT, \_Traits >::open(), std::basic\_filebuf< \_CharT, \_Traits >::overflow(), std::basic\_filebuf< \_CharT, \_Traits >::pbackfail(), std::basic\_filebuf< \_CharT, \_Traits >::showmanyc(), \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >::stdio\_filebuf(), std::basic\_filebuf< \_CharT, \_Traits >::underflow(), std::basic\_filebuf< \_CharT, \_Traits >::xsgetn(), and std::basic\_filebuf< \_CharT, \_Traits >::xsputn().

**5.376.5.11** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_out_beg  
[protected, inherited]`

Entry point for [imbue\(\)](#).

**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

Calls the derived imbue(loc).

Definition at line 185 of file streambuf.

Referenced by std::basic\_streambuf< char\_type, traits\_type >::pbase(), and std::basic\_streambuf< char\_type, traits\_type >::setp().

**5.376.5.12** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_out_cur  
[protected, inherited]`

Entry point for [imbue\(\)](#).

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived imbue(loc).

Definition at line 186 of file streambuf.

Referenced by std::basic\_streambuf< char\_type, traits\_type >::pbump(), std::basic\_streambuf< char\_type, traits\_type >::pptr(), and std::basic\_streambuf< char\_type, traits\_type >::setp().

**5.376.5.13** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_out_end  
[protected, inherited]`

Entry point for [imbue\(\)](#).

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived imbue(loc).

Definition at line 187 of file streambuf.

Referenced by std::basic\_streambuf< char\_type, traits\_type >::epptr(), and std::basic\_streambuf< char\_type, traits\_type >::setp().

**5.376.5.14** `template<typename _CharT, typename _Traits> char_type  
std::basic_filebuf< _CharT, _Traits >::_M_pback [protected]`

Necessary bits for putback buffer management.

**Note**

pbacks of over one character are not currently supported.

Definition at line 139 of file fstream.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::\_M\_create\_pback().

**5.376.5.15** `template<typename _CharT, typename _Traits> char_type*  
std::basic_filebuf< _CharT, _Traits >::_M_pback_cur_save  
[protected]`

Necessary bits for putback buffer management.

**Note**

pbacks of over one character are not currently supported.

Definition at line 140 of file fstream.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::\_M\_create\_pback(), and std::basic\_filebuf< char\_type, traits\_type >::\_M\_destroy\_pback().

**5.376.5.16** `template<typename _CharT, typename _Traits> char_type*  
std::basic_filebuf< _CharT, _Traits >::_M_pback_end_save  
[protected]`

Necessary bits for putback buffer management.

**Note**

pbacks of over one character are not currently supported.

Definition at line 141 of file fstream.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::\_M\_create\_pback(), and std::basic\_filebuf< char\_type, traits\_type >::\_M\_destroy\_pback().

**5.376.5.17** `template<typename _CharT, typename _Traits> bool  
std::basic_filebuf< _CharT, _Traits >::_M_pback_init  
[protected]`

Necessary bits for putback buffer management.

#### Note

pbacks of over one character are not currently supported.

Definition at line 142 of file `fstream`.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_create_pback()`, `std::basic_filebuf< char_type, traits_type >::_M_destroy_pback()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

**5.376.5.18** `template<typename _CharT, typename _Traits> bool  
std::basic_filebuf< _CharT, _Traits >::_M_reading  
[protected]`

`_M_reading == false && _M_writing == false` for **uncommitted** mode; `_M_reading == true` for **read** mode; `_M_writing == true` for **write** mode;

NB: `_M_reading == true && _M_writing == true` is unused.

Definition at line 130 of file `fstream`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::open()`, `std::basic_filebuf< _CharT, _Traits >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_filebuf< _CharT, _Traits >::seekoff()`, `__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, `std::basic_filebuf< _CharT, _Traits >::xsgetn()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

The documentation for this class was generated from the following files:

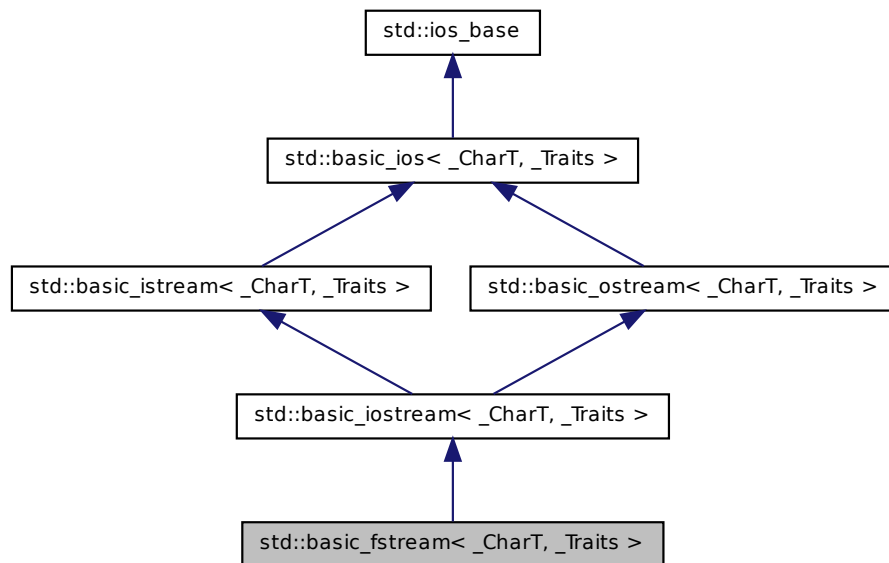
- [fstream](#)
- [fstream.tcc](#)

## 5.377 `std::basic_fstream<_CharT, _Traits>` Class Template Reference

Controlling input and output for files.

This class supports reading from and writing to named files, using the inherited functions from [std::basic\\_iostream](#). To control the associated sequence, an instance of [std::basic\\_filebuf](#) is used, which this page refers to as `sb`.

Inheritance diagram for std::basic\_fstream<\_CharT, \_Traits>:



### Public Types

- typedef `ctype<_CharT>` `__ctype_type`
- typedef `ctype<_CharT>` `__ctype_type`
- typedef `basic_filebuf<char_type, traits_type>` `__filebuf_type`
- typedef `basic_ios<_CharT, _Traits>` `__ios_type`
- typedef `basic_ios<char_type, traits_type>` `__ios_type`
- typedef `basic_iostream<char_type, traits_type>` `__iostream_type`
- typedef `basic_istream<_CharT, _Traits>` `__istream_type`
- typedef `basic_istream<_CharT, _Traits>` `__istream_type`
- typedef `num_get<_CharT, istreambuf_iterator<_CharT, _Traits>>` `__num_get_type`
- typedef `num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>>` `__num_put_type`
- typedef `basic_ostream<_CharT, _Traits>` `__ostream_type`
- typedef `basic_streambuf<_CharT, _Traits>` `__streambuf_type`
- typedef `basic_streambuf<_CharT, _Traits>` `__streambuf_type`

- typedef `_CharT` `char_type`
- typedef `_CharT` `char_type`
- enum `event` { `erase_event`, `imbue_event`, `copyfmt_event` }
- typedef void(\* `event_callback`)(`event`, `ios_base` &, int)
- typedef `_Ios_Fmtflags` `fmtflags`
- typedef `traits_type::int_type` `int_type`
- typedef `_Traits::int_type` `int_type`
- typedef int `io_state`
- typedef `_Ios_Iostate` `iostate`
- typedef `traits_type::off_type` `off_type`
- typedef `_Traits::off_type` `off_type`
- typedef int `open_mode`
- typedef `_Ios_Openmode` `openmode`
- typedef `_Traits::pos_type` `pos_type`
- typedef `traits_type::pos_type` `pos_type`
- typedef int `seek_dir`
- typedef `_Ios_Seekdir` `seekdir`
- typedef `std::streamoff` `streamoff`
- typedef `std::streampos` `streampos`
- typedef `_Traits` `traits_type`
- typedef `_Traits` `traits_type`
  
- typedef `num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>>` `num_put_type`

### Public Member Functions

- `basic_fstream` ()
- `basic_fstream` (const char \*\_\_s, `ios_base::openmode` \_\_mode=`ios_base::in|ios_base::out`)
- `basic_fstream` (const `std::string` &\_\_s, `ios_base::openmode` \_\_mode=`ios_base::in|ios_base::out`)
- `~basic_fstream` ()
- const `locale` & `_M_getloc` () const
- void `_M_setstate` (`iostate` \_\_state)
- bool `bad` () const
- void `clear` (`iostate` \_\_state=`goodbit`)
- void `close` ()
- `basic_ios` & `copyfmt` (const `basic_ios` &\_\_rhs)
- bool `eof` () const
- `iostate` `exceptions` () const
- void `exceptions` (`iostate` \_\_except)

- `bool fail () const`
- `char_type fill () const`
- `char_type fill (char_type __ch)`
- `fmtflags flags () const`
- `fmtflags flags (fmtflags __fmtfl)`
- `__ostream_type & flush ()`
- `streamsize gcount () const`
- `template<>`  
`basic_istream< wchar_t > & getline (char_type *__s, streamsize __n, char_type __delim)`
- `template<>`  
`basic_istream< char > & getline (char_type *__s, streamsize __n, char_type __delim)`
- `locale getloc () const`
- `bool good () const`
- `template<>`  
`basic_istream< wchar_t > & ignore (streamsize __n)`
- `template<>`  
`basic_istream< wchar_t > & ignore (streamsize __n, int_type __delim)`
- `template<>`  
`basic_istream< char > & ignore (streamsize __n)`
- `template<>`  
`basic_istream< char > & ignore (streamsize __n, int_type __delim)`
- `locale imbue (const locale &__loc)`
- `bool is_open ()`
- `bool is_open () const`
- `long & iword (int __ix)`
- `char narrow (char_type __c, char __dfault) const`
- `void open (const char *__s, ios_base::openmode __mode=ios_base::in|ios_base::out)`
- `void open (const std::string &__s, ios_base::openmode __mode=ios_base::in|ios_base::out)`
- `streamsize precision () const`
- `streamsize precision (streamsize __prec)`
- `void *& pword (int __ix)`
- `basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > *__sb)`
- `__filebuf_type * rdbuf () const`
- `iosstate rdstate () const`
- `void register_callback (event_callback __fn, int __index)`
- `__ostream_type & seekp (pos_type)`
- `__ostream_type & seekp (off_type, ios_base::seekdir)`
- `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`

- `fmtflags setf (fmtflags __fmtfl)`
  - `void setstate (iostate __state)`
  - `pos_type tellp ()`
  - `basic_ostream<_CharT, _Traits> * tie () const`
  - `basic_ostream<_CharT, _Traits> * tie (basic_ostream<_CharT, _Traits> * __tistr)`
  - `void unsetf (fmtflags __mask)`
  - `char_type widen (char __c) const`
  - `streamsize width (streamsize __wide)`
  - `streamsize width () const`
- 
- `__istream_type & operator>> (__istream_type &(__pf)(__istream_type &))`
  - `__istream_type & operator>> (__ios_type &(__pf)(__ios_type &))`
  - `__istream_type & operator>> (ios_base &(__pf)(ios_base &))`

### Arithmetic Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__istream_type & operator>> (bool &__n)`
- `__istream_type & operator>> (short &__n)`
- `__istream_type & operator>> (unsigned short &__n)`
- `__istream_type & operator>> (int &__n)`
- `__istream_type & operator>> (unsigned int &__n)`
- `__istream_type & operator>> (long &__n)`
- `__istream_type & operator>> (unsigned long &__n)`
- `__istream_type & operator>> (long long &__n)`
- `__istream_type & operator>> (unsigned long long &__n)`
- `__istream_type & operator>> (float &__f)`
- `__istream_type & operator>> (double &__f)`
- `__istream_type & operator>> (long double &__f)`
- `__istream_type & operator>> (void *&__p)`
- `__istream_type & operator>> (__streambuf_type * __sb)`



### Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `int_type get ()`
- `__istream_type & get (char_type &__c)`
- `__istream_type & get (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & get (char_type *__s, streamsize __n)`
- `__istream_type & get (__streambuf_type &__sb, char_type __delim)`
- `__istream_type & get (__streambuf_type &__sb)`
- `__istream_type & getline (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & getline (char_type *__s, streamsize __n)`
- `__istream_type & ignore ()`
- `__istream_type & ignore (streamsize __n)`
- `__istream_type & ignore (streamsize __n, int_type __delim)`
- `int_type peek ()`
- `__istream_type & read (char_type *__s, streamsize __n)`
- `streamsize readsome (char_type *__s, streamsize __n)`
- `__istream_type & putback (char_type __c)`
- `__istream_type & unget ()`
- `int sync ()`
- `pos_type tellg ()`
- `__istream_type & seekg (pos_type)`
- `__istream_type & seekg (off_type, ios_base::seekdir)`
- `operator void * () const`
- `bool operator! () const`
- `__ostream_type & operator<< (__ostream_type &(__pf)(__ostream_type &))`
- `__ostream_type & operator<< (__ios_type &(__pf)(__ios_type &))`
- `__ostream_type & operator<< (ios_base &(__pf)(ios_base &))`

### Arithmetic Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__ostream_type & operator<< (long __n)`
- `__ostream_type & operator<< (unsigned long __n)`
- `__ostream_type & operator<< (bool __n)`
- `__ostream_type & operator<< (short __n)`
- `__ostream_type & operator<< (unsigned short __n)`
- `__ostream_type & operator<< (int __n)`
- `__ostream_type & operator<< (unsigned int __n)`
- `__ostream_type & operator<< (long long __n)`
- `__ostream_type & operator<< (unsigned long long __n)`
- `__ostream_type & operator<< (double __f)`
- `__ostream_type & operator<< (float __f)`
- `__ostream_type & operator<< (long double __f)`
- `__ostream_type & operator<< (const void *__p)`
- `__ostream_type & operator<< (__streambuf_type *__sb)`

### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type & put (char_type __c)`
- `void _M_write (const char_type *__s, streamsize __n)`
- `__ostream_type & write (const char_type *__s, streamsize __n)`

### Static Public Member Functions

- static bool `sync_with_stdio` (bool \_\_sync=true)
- static int `xalloc` () throw ()

### Static Public Attributes

- static const `fmtflags adjustfield`
- static const `openmode app`

- static const `openmode ate`
- static const `iostate badbit`
- static const `fmtflags basefield`
- static const `seekdir beg`
- static const `openmode binary`
- static const `fmtflags boolalpha`
- static const `seekdir cur`
- static const `fmtflags dec`
- static const `seekdir end`
- static const `iostate eofbit`
- static const `iostate failbit`
- static const `fmtflags fixed`
- static const `fmtflags floatfield`
- static const `iostate goodbit`
- static const `fmtflags hex`
- static const `openmode in`
- static const `fmtflags internal`
- static const `fmtflags left`
- static const `fmtflags oct`
- static const `openmode out`
- static const `fmtflags right`
- static const `fmtflags scientific`
- static const `fmtflags showbase`
- static const `fmtflags showpoint`
- static const `fmtflags showpos`
- static const `fmtflags skipws`
- static const `openmode trunc`
- static const `fmtflags unitbuf`
- static const `fmtflags uppercase`

### Protected Types

- enum { `_S_local_word_size` }

### Protected Member Functions

- void `_M_cache_locale` (const `locale` &\_\_loc)
- void `_M_call_callbacks` (`event` \_\_ev) throw ()
- void `_M_dispose_callbacks` (void) throw ()
- template<typename \_ValueT >  
  `__istream_type` & `_M_extract` (\_ValueT &\_\_v)
- `_Words` & `_M_grow_words` (int \_\_index, bool \_\_iword)

- `void _M_init () throw ()`
- `template<typename _ValueT >  
__ostream_type & _M_insert (_ValueT __v)`
- `void init (basic_streambuf<_CharT, _Traits> *__sb)`

#### Protected Attributes

- `_Callback_list * _M_callbacks`
- `const __ctype_type * _M_ctype`
- `iosstate _M_exception`
- `char_type _M_fill`
- `bool _M_fill_init`
- `fmtflags _M_flags`
- `streamsize _M_gcount`
- `locale _M_ios_locale`
- `_Words _M_local_word [_S_local_word_size]`
- `const __num_get_type * _M_num_get`
- `const __num_put_type * _M_num_put`
- `streamsize _M_precision`
- `basic_streambuf<_CharT, _Traits> * _M_streambuf`
- `iosstate _M_streambuf_state`
- `basic_ostream<_CharT, _Traits> * _M_tie`
- `streamsize _M_width`
- `_Words * _M_word`
- `int _M_word_size`
- `_Words _M_word_zero`

#### Friends

- `class sentry`
- `class sentry`

#### 5.377.1 Detailed Description

`template<typename _CharT, typename _Traits> class std::basic_fstream<_CharT, _Traits>`

Controlling input and output for files.

This class supports reading from and writing to named files, using the inherited functions from `std::basic_ostream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.

Definition at line 761 of file `fstream`.

### 5.377.2 Member Typedef Documentation

**5.377.2.1** `template<typename _CharT, typename _Traits> typedef  
ctype<_CharT> std::basic_istream<_CharT, _Traits  
>::__ctype_type [inherited]`

These are non-standard types.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Definition at line 73 of file `istream`.

**5.377.2.2** `template<typename _CharT, typename _Traits> typedef  
ctype<_CharT> std::basic_ostream<_CharT, _Traits  
>::__ctype_type [inherited]`

These are non-standard types.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Definition at line 73 of file `ostream`.

**5.377.2.3** `template<typename _CharT, typename _Traits> typedef  
num_get<_CharT, istreambuf_iterator<_CharT, _Traits>  
> std::basic_istream<_CharT, _Traits>::__num_get_type  
[inherited]`

These are non-standard types.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Definition at line 72 of file `istream`.

**5.377.2.4** `template<typename _CharT, typename _Traits> typedef  
num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>  
> std::basic_ostream<_CharT, _Traits>::__num_put_type  
[inherited]`

These are non-standard types.

Reimplemented in `std::basic_ostream<_CharT, _Traits>`, `std::basic_ostream<char, _Traits>`, and `std::basic_ostream<char>`.

Definition at line 86 of file `basic_ios.h`.

**5.377.2.5** `template<typename _CharT, typename _Traits> typedef  
num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>  
> std::basic_ostream<_CharT, _Traits>::__num_put_type  
[inherited]`

These are non-standard types.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Definition at line 72 of file `ostream`.

**5.377.2.6** `template<typename _CharT, typename _Traits> typedef _CharT  
std::basic_istream<_CharT, _Traits>::char_type [inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Reimplemented in `std::basic_ifstream<_CharT, _Traits>`, and `std::basic_istreamstream<_CharT, _Traits, _Alloc>`.

Definition at line 61 of file `istream`.

**5.377.2.7** `template<typename _CharT, typename _Traits> typedef _CharT  
std::basic_fstream<_CharT, _Traits>::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_iostream<_CharT, _Traits>`.

Definition at line 765 of file `fstream`.

**5.377.2.8** `typedef void(* std::ios_base::event_callback)(event, ios_base &, int)  
[inherited]`

The type of an event callback function.

#### Parameters

*event* One of the members of the event enum.

*ios\_base* Reference to the `ios_base` object.

*int* The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several [ios\\_base](#) and [basic\\_ios](#) functions, specifically [imbue\(\)](#), [copyfmt\(\)](#), and [~ios\(\)](#).

Definition at line 438 of file [ios\\_base.h](#).

#### **5.377.2.9 typedef \_Ios\_Fmtflags std::ios\_base::fmtflags [inherited]**

This is a bitmask type.

*\_Ios\_Fmtflags* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type *fmtflags* are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 257 of file [ios\\_base.h](#).

**5.377.2.10** `template<typename _CharT, typename _Traits> typedef  
_Traits::int_type std::basic_istream<_CharT, _Traits>::int_type  
[inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Reimplemented in `std::basic_ifstream<_CharT, _Traits>`, and `std::basic_istreamstream<_CharT, _Traits, _Alloc>`.

Definition at line 62 of file `istream`.

**5.377.2.11** `template<typename _CharT, typename _Traits> typedef  
traits_type::int_type std::basic_fstream<_CharT, _Traits  
>::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_iostream<_CharT, _Traits>`.

Definition at line 767 of file `fstream`.

**5.377.2.12** `typedef _Ios_Iostate std::ios_base::iostate [inherited]`

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 332 of file `ios_base.h`.



**5.377.2.13** `template<typename _CharT, typename _Traits> typedef  
_Traits::off_type std::basic_istream<_CharT, _Traits>::off_type  
[inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Reimplemented in `std::basic_ifstream<_CharT, _Traits>`, and `std::basic_istream<_CharT, _Traits, _Alloc>`.

Definition at line 64 of file `istream`.

**5.377.2.14** `template<typename _CharT, typename _Traits> typedef  
traits_type::off_type std::basic_fstream<_CharT, _Traits>::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_iostream<_CharT, _Traits>`.

Definition at line 769 of file `fstream`.

**5.377.2.15** `typedef _Ios_Openmode std::ios_base::openmode [inherited]`

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 363 of file `ios_base.h`.

**5.377.2.16** `template<typename _CharT, typename _Traits> typedef traits_type::pos_type std::basic_fstream<_CharT, _Traits>::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_istream<_CharT, _Traits>`.

Definition at line 768 of file `fstream`.

**5.377.2.17** `template<typename _CharT, typename _Traits> typedef _Traits::pos_type std::basic_istream<_CharT, _Traits>::pos_type [inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Reimplemented in `std::basic_ifstream<_CharT, _Traits>`, and `std::basic_istream<_CharT, _Traits, _Alloc>`.

Definition at line 63 of file `istream`.

**5.377.2.18** `typedef _Ios_Seekdir std::ios_base::seekdir [inherited]`

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 395 of file `ios_base.h`.

**5.377.2.19** `template<typename _CharT, typename _Traits> typedef _Traits std::basic_fstream<_CharT, _Traits>::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_iostream<\\_CharT, \\_Traits>](#).

Definition at line 766 of file fstream.

**5.377.2.20** `template<typename _CharT, typename _Traits> typedef  
_Traits std::basic_istream<_CharT, _Traits>::traits_type  
[inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_ios<\\_CharT, \\_Traits>](#).

Reimplemented in [std::basic\\_ifstream<\\_CharT, \\_Traits>](#), and [std::basic\\_istream<\\_CharT, \\_Traits, \\_Alloc>](#).

Definition at line 65 of file istream.

### **5.377.3 Member Enumeration Documentation**

**5.377.3.1** `enum std::ios_base::event [inherited]`

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during [imbue\(\)](#). `copyfmt_event` is used during `copyfmt()`.

Definition at line 421 of file ios\_base.h.

### **5.377.4 Constructor & Destructor Documentation**

**5.377.4.1** `template<typename _CharT, typename _Traits>  
std::basic_fstream<_CharT, _Traits>::basic_fstream ( )  
[inline]`

Default constructor.

Initializes `sb` using its default constructor, and passes `&sb` to the base class initializer. Does not open any files (you haven't given it a filename to open).

Definition at line 788 of file fstream.

References [std::basic\\_ios<\\_CharT, \\_Traits>::init\(\)](#).

**5.377.4.2** `template<typename _CharT , typename _Traits >  
std::basic_fstream< _CharT, _Traits >::basic_fstream ( const char *  
__s, ios_base::openmode __mode = ios_base::in | ios_base::out )  
[inline, explicit]`

Create an input/output file stream.

#### Parameters

*s* Null terminated string specifying the filename.

*mode* Open file in specified mode (see [std::ios\\_base](#)).

Tip: When using `std::string` to hold the filename, you must use `.c_str()` before passing it to this constructor.

Definition at line 801 of file `fstream`.

References `std::basic_ios<_CharT, _Traits>::init()`, and `std::basic_fstream<_CharT, _Traits>::open()`.

**5.377.4.3** `template<typename _CharT , typename _Traits >  
std::basic_fstream< _CharT, _Traits >::basic_fstream  
( const std::string & __s, ios_base::openmode __mode =  
ios_base::in | ios_base::out ) [inline, explicit]`

Create an input/output file stream.

#### Parameters

*s* Null terminated string specifying the filename.

*mode* Open file in specified mode (see [std::ios\\_base](#)).

Definition at line 816 of file `fstream`.

References `std::basic_ios<_CharT, _Traits>::init()`, and `std::basic_fstream<_CharT, _Traits>::open()`.

**5.377.4.4** `template<typename _CharT , typename _Traits >  
std::basic_fstream< _CharT, _Traits >::~basic_fstream ( )  
[inline]`

The destructor does nothing.

The file is closed by the filebuf object, not the formatting stream.

Definition at line 831 of file fstream.

### **5.377.5 Member Function Documentation**

#### **5.377.5.1 const locale& std::ios\_base::\_M\_getloc ( ) const [inline, inherited]**

Locale access.

##### **Returns**

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 708 of file ios\_base.h.

Referenced by std::money\_get<\_CharT, \_InIter>::do\_get(), std::num\_get<\_CharT, \_InIter>::do\_get(), std::time\_get<\_CharT, \_InIter>::do\_get\_date(), std::time\_get<\_CharT, \_InIter>::do\_get\_monthname(), std::time\_get<\_CharT, \_InIter>::do\_get\_time(), std::time\_get<\_CharT, \_InIter>::do\_get\_weekday(), std::time\_get<\_CharT, \_InIter>::do\_get\_year(), std::time\_put<\_CharT, \_OutIter>::do\_put(), std::num\_put<\_CharT, \_OutIter>::do\_put(), and std::time\_put<\_CharT, \_OutIter>::put().

#### **5.377.5.2 template<typename \_CharT, typename \_Traits> void std::basic\_ostream<\_CharT, \_Traits>::\_M\_write ( const char\_type \* \_\_s, streamsize \_\_n ) [inline, inherited]**

Simple insertion.

##### **Parameters**

*c* The character to insert.

##### **Returns**

\*this

Tries to insert *c*.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 289 of file ostream.

Referenced by `std::basic_ostream<_CharT, _Traits>::write()`.

**5.377.5.3 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::bad ( ) const [inline, inherited]`**

Fast error checking.

**Returns**

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 203 of file basic\_ios.h.

**5.377.5.4 `template<typename _CharT, typename _Traits> void std::basic_ios<_CharT, _Traits>::clear ( iostate __state = goodbit ) [inherited]`**

[Re]sets the error state.

**Parameters**

*state* The new state flag(s) to set.

See [std::ios\\_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

Definition at line 42 of file basic\_ios.tcc.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::exceptions()`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::rdstate()`.

Referenced by `std::basic_ios<char, _Traits>::exceptions()`, `std::basic_fstream<_CharT, _Traits>::open()`, `std::basic_ofstream<_CharT, _Traits>::open()`, `std::basic_ifstream<_CharT, _Traits>::open()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ios<char, _Traits>::setstate()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

**5.377.5.5** `template<typename _CharT, typename _Traits > void  
std::basic_fstream< _CharT, _Traits >::close ( ) [inline]`

Close the file.

Calls `std::basic_filebuf::close()`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 911 of file `fstream`.

References `std::basic_filebuf< _CharT, _Traits >::close()`, `std::ios_base::failbit`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.377.5.6** `template<typename _CharT, typename _Traits > basic_ios<  
_CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (  
const basic_ios< _CharT, _Traits > & __rhs ) [inherited]`

Copies fields of `__rhs` into this.

#### Parameters

`__rhs` The source values for the copies.

#### Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy `exceptions()`.

Definition at line 64 of file `basic_ios.tcc`.

References `std::basic_ios< _CharT, _Traits >::exceptions()`, `std::basic_ios< _CharT, _Traits >::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios< _CharT, _Traits >::tie()`, and `std::ios_base::width()`.

**5.377.5.7** `template<typename _CharT, typename _Traits> bool std::basic_ios<  
_CharT, _Traits >::eof ( ) const [inline, inherited]`

Fast error checking.

**Returns**

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 182 of file `basic_ios.h`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

**5.377.5.8** `template<typename _CharT, typename _Traits> iostate  
std::basic_ios<_CharT, _Traits>::exceptions( ) const [inline,  
inherited]`

Throwing exceptions on errors.

**Returns**

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of [exceptions\(iostate\)](#) for the meaning of the return value.

Definition at line 214 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::clear()`, and `std::basic_ios<_CharT, _Traits>::copyfmt()`.

**5.377.5.9** `template<typename _CharT, typename _Traits> void std::basic_ios<  
_CharT, _Traits>::exceptions( iostate __except ) [inline,  
inherited]`

Throwing exceptions on errors.

**Parameters**

*except* The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type [std::ios\\_base::failure](#) is thrown.



If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
 std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

 std::ifstream f ("/etc/motd");

 std::cerr << "Setting badbit\n";
 f.setstate (std::ios_base::badbit);

 std::cerr << "Setting exception mask\n";
 f.exceptions (std::ios_base::badbit);
}
```

Definition at line 249 of file `basic_ios.h`.

**5.377.5.10** `template<typename _CharT, typename _Traits> bool  
std::basic_ios<_CharT, _Traits>::fail ( ) const [inline,  
inherited]`

Fast error checking.

#### Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Definition at line 193 of file `basic_ios.h`.

Referenced by `std::basic_ios<char, _Traits>::operator void*()`, `std::basic_ios<char, _Traits>::operator!()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_ostream<_CharT, _Traits>::tellp()`, and `std::regex_traits<_Ch_type>::value()`.

**5.377.5.11** `template<typename _CharT, typename _Traits> char_type  
std::basic_ios<_CharT, _Traits>::fill ( char_type __ch )  
[inline, inherited]`

Sets a new *empty* character.

#### Parameters

*ch* The new character.

#### Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 382 of file `basic_ios.h`.

**5.377.5.12** `template<typename _CharT, typename _Traits> char_type  
std::basic_ios<_CharT, _Traits>::fill ( ) const [inline,  
inherited]`

Retrieves the *empty* character.

#### Returns

The current fill character.

It defaults to a space ( ' ') in the current locale.

Definition at line 362 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, and `std::basic_ios<char, _Traits>::fill()`.

**5.377.5.13** `fmtflags std::ios_base::flags ( fmtflags __fmtfl ) [inline,  
inherited]`

Setting new format flags all at once.

#### Parameters

*fmtfl* The new flags to set.

#### Returns

The previous format control flags.

This function overwrites all the format flags with *fmtfl*.

Definition at line 564 of file ios\_base.h.

#### 5.377.5.14 fmtflags std::ios\_base::flags ( ) const [inline, inherited]

Access to format flags.

##### Returns

The format control flags for both input and output.

Definition at line 553 of file ios\_base.h.

Referenced by std::basic\_ios<\_CharT, \_Traits>::copyfmt(), std::num\_get<\_CharT, \_InIter>::do\_get(), std::num\_put<\_CharT, \_OutIter>::do\_put(), std::basic\_ostream<\_CharT, \_Traits>::operator<<(), std::operator<<(), std::operator>>(), and std::basic\_istream<\_CharT, \_Traits>::sentry::sentry().

#### 5.377.5.15 template<typename \_CharT, typename \_Traits> basic\_ostream<\_CharT, \_Traits> & std::basic\_ostream<\_CharT, \_Traits>::flush ( ) [inherited]

Synchronizing the stream buffer.

##### Returns

\*this

If *rdbuf()* is a null pointer, changes nothing.

Otherwise, calls *rdbuf()* -> *pubsync()*, and if that returns -1, sets *badbit*.

Definition at line 213 of file ostream.tcc.

References std::ios\_base::badbit, std::ios\_base::goodbit, std::basic\_ios<\_CharT, \_Traits>::rdbuf(), and std::basic\_ios<\_CharT, \_Traits>::setstate().

Referenced by std::flush().

#### 5.377.5.16 template<typename \_CharT, typename \_Traits> streamsize std::basic\_istream<\_CharT, \_Traits>::gcount ( ) const [inline, inherited]

Character counting.

#### Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 251 of file istream.

**5.377.5.17** `template<typename _CharT, typename _Traits > basic_istream<  
_CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits  
>::get ( void ) [inherited]`

Simple extraction.

#### Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 238 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.377.5.18** `template<typename _CharT, typename _Traits > basic_istream<  
_CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (   
char_type & __c ) [inherited]`

Simple extraction.

#### Parameters

*c* The character in which to store data.

#### Returns

`*this`

Tries to extract a character and store it in *c*. If none are available, sets failbit and returns `traits::eof()`.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 274 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.377.5.19** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get ( char_type * __s, streamsize __n, char_type __delim )`  
**[inherited]**

Simple multiple-character extraction.

**Parameters**

- s* Pointer to an array.
- n* Maximum number of characters to store in *s*.
- delim* A "stop" character.

**Returns**

\*this

Characters are extracted and stored into *s* until one of the following happens:

- *n*−1 characters are stored
- the input sequence reaches EOF
- the next character equals *delim*, in which case the character is not extracted

If no characters are stored, `failbit` is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 311 of file `istream.tcc`.

References std::basic\_istream<\_CharT, \_Traits>::\_M\_gcount, std::ios\_base::badbit, std::basic\_ios<\_CharT, \_Traits>::eof(), std::ios\_base::eofbit, std::ios\_base::failbit, std::ios\_base::goodbit, std::basic\_ios<\_CharT, \_Traits>::rdbuf(), std::basic\_ios<\_CharT, \_Traits>::setstate(), std::basic\_streambuf<\_CharT, \_Traits>::sgetc(), and std::basic\_streambuf<\_CharT, \_Traits>::snextc().

**5.377.5.20** template<typename \_CharT, typename \_Traits> basic\_istream<\_CharT, \_Traits> & std::basic\_istream<\_CharT, \_Traits>::get ( \_\_streambuf\_type & \_\_sb, char\_type \_\_delim ) [inherited]

Extraction into another streambuf.

#### Parameters

*sb* A streambuf in which to store data.

*delim* A "stop" character.

#### Returns

\*this

Characters are extracted and inserted into *sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 358 of file istream.tcc.

References std::basic\_istream<\_CharT, \_Traits>::\_M\_gcount, std::ios\_base::badbit, std::basic\_ios<\_CharT, \_Traits>::eof(), std::ios\_base::eofbit, std::ios\_base::failbit, std::ios\_base::goodbit, std::basic\_ios<\_CharT, \_Traits>::rdbuf(), std::basic\_ios<\_CharT, \_Traits>::setstate(), std::basic\_streambuf<\_CharT, \_Traits>::sgetc(), std::basic\_streambuf<\_CharT, \_Traits>::snextc(), and std::basic\_streambuf<\_CharT, \_Traits>::sputc().

**5.377.5.21** `template<typename _CharT, typename _Traits> __istream_type&  
std::basic_istream<_CharT, _Traits>::get ( char_type * __s,  
streamsize __n ) [inline, inherited]`

Simple multiple-character extraction.

#### Parameters

- s* Pointer to an array.
- n* Maximum number of characters to store in *s*.

#### Returns

\*this

Returns `get(s,n,widen('\n'))`.

Definition at line 335 of file istream.

**5.377.5.22** `template<typename _CharT, typename _Traits> __istream_type&  
std::basic_istream<_CharT, _Traits>::get ( __streambuf_type &  
__sb ) [inline, inherited]`

Extraction into another streambuf.

#### Parameters

- sb* A streambuf in which to store data.

#### Returns

\*this

Returns `get(sb,widen('\n'))`.

Definition at line 368 of file istream.

**5.377.5.23** `template<typename _CharT, typename _Traits> __istream_type&  
std::basic_istream<_CharT, _Traits>::getline ( char_type * __s,  
streamsize __n ) [inline, inherited]`

String extraction.

**Parameters**

- s* A character array in which to store the data.
- n* Maximum number of characters to extract.

**Returns**

`*this`

Returns `getline(s,n,widen('\n'))`.

Definition at line 408 of file `istream`.

Referenced by `std::basic_istream<char>::getline()`.

**5.377.5.24** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::getline ( char_type * __s, streamsize __n, char_type __delim ) [inherited]`

String extraction.

**Parameters**

- s* A character array in which to store the data.
- n* Maximum number of characters to extract.
- delim* A "stop" character.

**Returns**

`*this`

Extracts and stores characters into *s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals *delim*, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. *n*-1 characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)



In any case, a null character is stored in the next location in the array.

Definition at line 402 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sbumpc()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

#### 5.377.5.25 `std::ios_base::getloc()` `const` `[inline, inherited]`

Locale access.

##### Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 697 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::money_put<_CharT, _OutIter>::do_put()`, `std::basic_ios<_CharT, _Traits>::imbue()`, `std::operator>>()`, and `std::ws()`.

#### 5.377.5.26 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::good()` `const` `[inline, inherited]`

Fast error checking.

##### Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 172 of file `basic_ios.h`.

Referenced by `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

**5.377.5.27** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore( void ) [inherited]`

Discarding characters.

#### Parameters

*n* Number of characters to discard.

*delim* A "stop" character.

#### Returns

\*this

Extracts characters and throws them away until one of the following happens:

- if *n* != `std::numeric_limits<int>::max()`, *n* characters are extracted
- the input sequence reaches end-of-file
- the next character equals *delim* (in this case, the character is extracted); note that this condition will never occur if *delim* equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 462 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sbumpc()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.377.5.28** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore( streamsize __n ) [inherited]`

Simple extraction.

#### Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 495 of file istream.tcc.

References std::basic\_istream<\_CharT, \_Traits>::\_M\_gcount, std::ios\_base::badbit, std::basic\_ios<\_CharT, \_Traits>::eof(), std::ios\_base::eofbit, std::ios\_base::goodbit, std::basic\_ios<\_CharT, \_Traits>::rdbuf(), std::basic\_ios<\_CharT, \_Traits>::setstate(), std::basic\_streambuf<\_CharT, \_Traits>::sgetc(), and std::basic\_streambuf<\_CharT, \_Traits>::snextc().

**5.377.5.29** `template<typename _CharT, typename _Traits> basic_istream<  
_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore  
( streamsize __n, int_type __delim ) [inherited]`

Simple extraction.

#### Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 557 of file istream.tcc.

References std::basic\_istream<\_CharT, \_Traits>::\_M\_gcount, std::ios\_base::badbit, std::basic\_ios<\_CharT, \_Traits>::eof(), std::ios\_base::eofbit, std::ios\_base::goodbit, std::basic\_ios<\_CharT, \_Traits>::rdbuf(), std::basic\_streambuf<\_CharT, \_Traits>::sbumpc(), std::basic\_ios<\_CharT, \_Traits>::setstate(), std::basic\_streambuf<\_CharT, \_Traits>::sgetc(), and std::basic\_streambuf<\_CharT, \_Traits>::snextc().

**5.377.5.30** `template<typename _CharT, typename _Traits> locale  
std::basic_ios<_CharT, _Traits>::imbue ( const locale & __loc )  
[inherited]`

Moves to a new locale.

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Reimplemented from `std::ios_base`.

Definition at line 115 of file `basic_ios.tcc`.

References `std::ios_base::getloc()`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

Referenced by `std::operator<<()`.

**5.377.5.31** `template<typename _CharT, typename _Traits> void  
std::basic_ios<_CharT, _Traits>::init ( basic_streambuf<  
_CharT, _Traits> * __sb ) [protected, inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 127 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

Referenced by `std::basic_fstream<_CharT, _Traits>::basic_fstream()`, `std::basic_ifstream<_CharT, _Traits>::basic_ifstream()`, `std::basic_ios<char, _Traits>::basic_ios()`, `std::basic_istream<char>::basic_istream()`, `std::basic_istreamstream<_CharT, _Traits, _Alloc>::basic_istreamstream()`, `std::basic_ofstream<_CharT, _Traits>::basic_ofstream()`, `std::basic_ostream<char>::basic_ostream()`, `std::basic_ostringstream<_CharT, _Traits, _Alloc>::basic_ostringstream()`, and `std::basic_stringstream<_CharT, _Traits, _Alloc>::basic_stringstream()`.

**5.377.5.32** `template<typename _CharT, typename _Traits> bool  
std::basic_fstream<_CharT, _Traits>::is_open ( ) [inline]`

Wrapper to test for an open file.

### Returns

`rdbuf() -> is_open()`

Definition at line 850 of file `fstream`.

References `std::basic_filebuf<_CharT, _Traits>::is_open()`.

**5.377.5.33 long& std::ios\_base::iword ( int \_\_ix ) [inline, inherited]**

Access to integer array.

**Parameters**

*\_\_ix* Index into the array.

**Returns**

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 743 of file ios\_base.h.

**5.377.5.34 template<typename \_CharT, typename \_Traits> char  
std::basic\_ios<\_CharT, \_Traits>::narrow ( char\_type \_\_c, char  
\_\_dfault ) const [inline, inherited]**

Squeezes characters.

**Parameters**

*c* The character to narrow.

*dfault* The character to narrow.

**Returns**

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c,dfault)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.ht>

Definition at line 422 of file basic\_ios.h.

**5.377.5.35** `template<typename _CharT, typename _Traits> void  
std::basic_fstream<_CharT, _Traits>::open ( const char * __s,  
ios_base::openmode __mode = ios_base::in | ios_base::out )  
[inline]`

Opens an external file.

#### Parameters

- s* The name of the file.
- mode* The open mode flags.

Calls `std::basic_filebuf::open(s, mode)`. If that function fails, `failbit` is set in the stream's error state.

Tip: When using `std::string` to hold the filename, you must use `.c_str()` before passing it to this constructor.

Definition at line 871 of file `fstream`.

References `std::basic_ios<_CharT, _Traits>::clear()`, `std::ios_base::failbit`, `std::basic_filebuf<_CharT, _Traits>::open()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

Referenced by `std::basic_fstream<_CharT, _Traits>::basic_fstream()`.

**5.377.5.36** `template<typename _CharT, typename _Traits> void  
std::basic_fstream<_CharT, _Traits>::open ( const std::string &  
__s, ios_base::openmode __mode = ios_base::in | ios_base::out )  
[inline]`

Opens an external file.

#### Parameters

- s* The name of the file.
- mode* The open mode flags.

Calls `std::basic_filebuf::open(s, mode)`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 892 of file `fstream`.

References `std::basic_ios<_CharT, _Traits>::clear()`, `std::ios_base::failbit`, `std::basic_filebuf<_CharT, _Traits>::open()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.377.5.37** `template<typename _CharT, typename _Traits> std::basic_ios<_CharT, _Traits >::operator void * ( ) const [inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 113 of file `basic_ios.h`.

**5.377.5.38** `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits >::operator! ( ) const [inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 117 of file `basic_ios.h`.

**5.377.5.39** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits >::operator<< ( unsigned long __n ) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 171 of file `ostream`.

**5.377.5.40** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream<_CharT, _Traits>::operator<<( unsigned  
short __n ) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 182 of file ostream.

**5.377.5.41** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream<_CharT, _Traits>::operator<<( double __f  
) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 211 of file ostream.

**5.377.5.42** `template<typename _CharT, typename _Traits> basic_ostream<  
_CharT, _Traits> & std::basic_ostream<_CharT, _Traits  
>::operator<<( short __n ) [inherited]`

Basic arithmetic inserters.



**Parameters**

A variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 94 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

**5.377.5.43** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream<_CharT, _Traits>::operator<< (  
__ostream_type &(*)(__ostream_type &) __pf ) [inline,  
inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the `iomanip` header.

Definition at line 110 of file ostream.

**5.377.5.44** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream<_CharT, _Traits>::operator<< ( __ios_type  
&(*)(__ios_type &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the `iomanip` header.

Definition at line 119 of file ostream.

**5.377.5.45** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream<_CharT, _Traits>::operator<< ( ios_base  
&(*)(ios_base &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

Definition at line 129 of file `ostream`.

**5.377.5.46** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream<_CharT, _Traits>::operator<< ( long __n )  
[inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 167 of file `ostream`.

**5.377.5.47** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream<_CharT, _Traits>::operator<< ( bool __n )  
[inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 175 of file `ostream`.

**5.377.5.48** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream<_CharT, _Traits>::operator<<( unsigned int  
__n ) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 193 of file ostream.

**5.377.5.49** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream<_CharT, _Traits>::operator<<( float __f )  
[inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 215 of file ostream.

**5.377.5.50** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream<_CharT, _Traits>::operator<<( long double  
__f ) [inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 223 of file `ostream`.

**5.377.5.51** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( const void * __p ) [inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 227 of file `ostream`.

**5.377.5.52** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<<( int __n ) [inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the [num\\_get](#) facet) to perform numeric formatting.

Definition at line 108 of file ostream.tcc.

References [std::ios\\_base::basefield](#), [std::ios\\_base::flags\(\)](#), [std::ios\\_base::hex](#), and [std::ios\\_base::oct](#).

**5.377.5.53** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream<_CharT, _Traits>::operator<< ( long long  
__n ) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the [num\\_get](#) facet) to perform numeric formatting.

Definition at line 202 of file ostream.

**5.377.5.54** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream<_CharT, _Traits>::operator<< ( unsigned  
long long __n ) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the [num\\_get](#) facet) to perform numeric formatting.

Definition at line 206 of file ostream.

**5.377.5.55** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<<( __streambuf_type * __sb ) [inherited]`

Extracting from another streambuf.

#### Parameters

*sb* A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from *sb* and inserted into *\*this* until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from *sb*, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 122 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.377.5.56** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>>( __ios_type &(*)(__ios_type &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 126 of file istream.

**5.377.5.57** `template<typename _CharT, typename _Traits> __istream_type&  
std::basic_istream<_CharT, _Traits>::operator>> ( unsigned  
long & __n ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 191 of file istream.

**5.377.5.58** `template<typename _CharT, typename _Traits> __istream_type&  
std::basic_istream<_CharT, _Traits>::operator>> ( long long &  
__n ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 196 of file istream.

**5.377.5.59** `template<typename _CharT, typename _Traits> __istream_type&  
std::basic_istream<_CharT, _Traits>::operator>> ( unsigned  
long long & __n ) [inline, inherited]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 200 of file `istream`.

**5.377.5.60** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( ios_base &(*) (ios_base &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iomanip` header.

Definition at line 133 of file `istream`.

**5.377.5.61** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( float & __f ) [inline, inherited]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 205 of file `istream`.



**5.377.5.62** `template<typename _CharT, typename _Traits> __istream_type&  
std::basic_istream<_CharT, _Traits>::operator>> ( unsigned int  
& __n ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 183 of file `istream`.

**5.377.5.63** `template<typename _CharT, typename _Traits> __istream_type&  
std::basic_istream<_CharT, _Traits>::operator>> ( double &  
__f ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 209 of file `istream`.

**5.377.5.64** `template<typename _CharT, typename _Traits> __istream_type&  
std::basic_istream<_CharT, _Traits>::operator>> ( bool & __n  
) [inline, inherited]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 169 of file `istream`.

**5.377.5.65** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( long double & __f ) [inline, inherited]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 213 of file `istream`.

**5.377.5.66** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::operator>> ( short & __n ) [inherited]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 116 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get<_CharT, _InIter>::get()`, `std::ios_base::goodbit`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.377.5.67** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::operator>> ( __streambuf_type * __sb ) [inherited]`

Extracting into another streambuf.

#### Parameters

*sb* A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the *sb* streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 206 of file `istream.tcc`.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.377.5.68** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( __istream_type &(*)(__istream_type &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 122 of file `istream`.

```
5.377.5.69 template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream<_CharT, _Traits>::operator>> (unsigned
short & __n) [inline, inherited]
```

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 176 of file `istream`.

```
5.377.5.70 template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream<_CharT, _Traits>::operator>> (long & __n
) [inline, inherited]
```

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 187 of file `istream`.

**5.377.5.71** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::operator>> ( int & __n ) [inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 161 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get<_CharT, _InIter>::get()`, `std::ios_base::goodbit`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.377.5.72** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( void *& __p ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 217 of file `istream`.

**5.377.5.73** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits>::peek ( void ) [inherited]`

Looking ahead in the stream.

### Returns

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 622 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

#### 5.377.5.74 `streamsize std::ios_base::precision ( ) const [inline, inherited]`

Flags access.

### Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 623 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, and `std::operator<<()`.

#### 5.377.5.75 `streamsize std::ios_base::precision ( streamsize __prec ) [inline, inherited]`

Changing flags.

### Parameters

*prec* The new precision value.

### Returns

The previous value of `precision()`.

Definition at line 632 of file `ios_base.h`.

**5.377.5.76** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::put (char_type __c) [inherited]`

Simple insertion.

#### Parameters

*c* The character to insert.

#### Returns

\*this

Tries to insert *c*.

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 151 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

Referenced by `std::endl()`, and `std::ends()`.

**5.377.5.77** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::putback (char_type __c) [inherited]`

Unextracting a single character.

#### Parameters

*c* The character to push back into the input stream.

#### Returns

\*this

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

**Note**

Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 713 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::clear()`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::rdstate()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sputbackc()`.

Referenced by `std::operator>>()`.

**5.377.5.78** `void*& std::ios_base::pword ( int __ix ) [inline, inherited]`

Access to void pointer array.

**Parameters**

`__ix` Index into the array.

**Returns**

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 764 of file `ios_base.h`.

**5.377.5.79** `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rdbuf ( basic_streambuf<_CharT, _Traits> * __sb ) [inherited]`

Changing the underlying buffer.



**Parameters**

*sb* The new stream buffer.

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream foo; // or some other derived type
std::streambuf* p =;

foo.ios::rdbuf(p); // ios == basic_ios<char>
```

Definition at line 54 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::clear()`.

**5.377.5.80** `template<typename _CharT, typename _Traits> __filebuf_type*`  
`std::basic_fstream<_CharT, _Traits>::rdbuf( ) const`  
`[inline]`

Accessing the underlying buffer.

**Returns**

The current `basic_filebuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Definition at line 842 of file `fstream`.

**5.377.5.81** `template<typename _CharT, typename _Traits> iostate`  
`std::basic_ios<_CharT, _Traits>::rdstate( ) const [inline,`  
`inherited]`

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 129 of file `basic_ios.h`.

Referenced by `std::basic_ios<char, _Traits>::bad()`, `std::basic_ios<_CharT, _Traits>::clear()`, `std::basic_ios<char, _Traits>::eof()`, `std::basic_ios<char, _Traits>::fail()`, `std::basic_ios<char, _Traits>::good()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ios<char, _Traits>::setstate()`, and `std::basic_istream<_CharT, _Traits>::unset()`.

**5.377.5.82** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::read (char_type * __s, streamsize __n) [inherited]`

Extraction without delimiters.

**Parameters**

- s* A character array.
- n* Maximum number of characters to store.

**Returns**

\*this

If the stream state is `good()`, extracts characters and stores them into *s* until one of the following happens:

- *n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 652 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.377.5.83** `template<typename _CharT, typename _Traits > streamsize  
std::basic_istream< _CharT, _Traits >::readsome ( char_type *  
__s, streamsize __n ) [inherited]`

Extraction until the buffer is exhausted, but no more.

#### Parameters

- s* A character array.
- n* Maximum number of characters to store.

#### Returns

The number of characters extracted.

Extracts characters and stores them into *s* depending on the number of characters remaining in the streambuf's buffer, `rdbuf() -> in_avail()`, called *A* here:

- if *A* == -1, sets eofbit and extracts no characters
- if *A* == 0, extracts no characters
- if *A* > 0, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 681 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::min()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.377.5.84** `void std::ios_base::register_callback ( event_callback __fn, int  
__index ) [inherited]`

Add the callback *\_\_fn* with parameter *\_\_index*.

#### Parameters

- \_\_fn* The function to add.
- \_\_index* The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**5.377.5.85** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg( off_type __off, ios_base::seekdir __dir ) [inherited]`

Changing the current read position.

#### Parameters

*off* A file offset object.

*dir* The direction in which to seek.

#### Returns

\*this

If `fail()` is not true, calls `rddbuf()` -> `pubseekoff(off, dir)`. If that function fails, sets failbit.

#### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 886 of file istream.tcc.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::clear()`, `std::ios_base::eofbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios<_CharT, _Traits>::rddbuf()`, `std::basic_ios<_CharT, _Traits>::rdstate()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.377.5.86** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg( pos_type __pos ) [inherited]`

Changing the current read position.

#### Parameters

*pos* A file position object.

#### Returns

\*this

If `fail()` is not true, calls `rdbuf() -> pubseekpos(pos)`. If that function fails, sets failbit.

#### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 847 of file istream.tcc.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::clear()`, `std::ios_base::eofbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::rdstate()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.377.5.87** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp( pos_type __pos ) [inherited]`

Changing the current write position.

#### Parameters

*pos* A file position object.

#### Returns

\*this

If `fail()` is not true, calls `rdbuf() -> pubseekpos(pos)`. If that function fails, sets failbit.

Definition at line 260 of file ostream.tcc.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.377.5.88** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp( off_type __off, ios_base::seekdir __dir ) [inherited]`

Changing the current write position.

**Parameters**

*off* A file offset object.

*dir* The direction in which to seek.

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

Definition at line 292 of file ostream.tcc.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.377.5.89** `fmtflags std::ios_base::setf ( fmtflags __fmtfl ) [inline, inherited]`

Setting new format flags.

**Parameters**

*fmtfl* Additional flags to set.

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 580 of file ios\_base.h.

Referenced by `std::dec()`, `std::fixed()`, `std::hex()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

**5.377.5.90** `fmtflags std::ios_base::setf ( fmtflags __fmtfl, fmtflags __mask ) [inline, inherited]`

Setting new format flags.

**Parameters**

*fmtfl* Additional flags to set.

*mask* The flags mask for *fmtfl*.

**Returns**

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 597 of file `ios_base.h`.

**5.377.5.91** `template<typename _CharT, typename _Traits> void  
std::basic_ios<_CharT, _Traits>::setstate ( iostate __state )  
[inline, inherited]`

Sets additional flags in the error state.

**Parameters**

*state* The additional state flag(s) to set.

See `std::ios_base::iostate` for the possible bit values.

Definition at line 149 of file `basic_ios.h`.

Referenced by `std::basic_ostream<char>::_M_write()`, `std::basic_fstream<_CharT, _Traits>::close()`, `std::basic_ofstream<_CharT, _Traits>::close()`, `std::basic_ifstream<_CharT, _Traits>::close()`, `std::basic_ostream<_CharT, _Traits>::flush()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_fstream<_CharT, _Traits>::open()`, `std::basic_ofstream<_CharT, _Traits>::open()`, `std::basic_ifstream<_CharT, _Traits>::open()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_ostream<_CharT, _Traits>::put()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::readsome()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, `std::basic_istream<_CharT, _Traits>::sync()`, `std::basic_istream<_CharT, _Traits>::unget()`, and `std::ws()`.

**5.377.5.92** `template<typename _CharT, typename _Traits> int  
std::basic_istream<_CharT, _Traits>::sync ( void )  
[inherited]`

Synchronizing the stream buffer.

#### Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

#### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 783 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_streambuf<_CharT, _Traits>::pubsync()`, `std::basic_istream<_CharT, _Traits>::rdbuf()`, and `std::basic_istream<_CharT, _Traits>::setstate()`.

**5.377.5.93** `static bool std::ios_base::sync_with_stdio ( bool __sync = true )  
[static, inherited]`

Interaction with the standard C I/O objects.

#### Parameters

*sync* Whether to synchronize or not.

#### Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>



**5.377.5.94** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::pos_type std::basic_istream<_CharT, _Traits>::tellg( void ) [inherited]`

Getting the current read position.

#### Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, in)`.

#### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 819 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::in`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

**5.377.5.95** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>::pos_type std::basic_ostream<_CharT, _Traits>::tellp( ) [inherited]`

Getting the current write position.

#### Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, out)`.

Definition at line 239 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::out`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

**5.377.5.96** `template<typename _CharT, typename _Traits>  
 basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits  
>::tie( ) const [inline, inherited]`

Fetches the current *tied* stream.

#### Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 287 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

**5.377.5.97** `template<typename _CharT, typename _Traits>  
 basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits  
>::tie( basic_ostream<_CharT, _Traits> * __tiestr ) [inline,  
 inherited]`

Ties this stream to an output stream.

#### Parameters

*tiestr* The output stream.

#### Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 299 of file `basic_ios.h`.

**5.377.5.98** `template<typename _CharT, typename _Traits> basic_istream<  
 _CharT, _Traits> & std::basic_istream<_CharT, _Traits>::unget  
( void ) [inherited]`

Unextracting the previous character.

**Returns**

\*this

If `rdbuf()` is not null, calls `rdbuf() -> sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

**Note**

Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 748 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::clear()`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::rdstate()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sungetc()`.

**5.377.5.99** `void std::ios_base::unsetf ( fmtflags __mask ) [inline, inherited]`

Clearing format flags.

**Parameters**

*mask* The flags to unset.

This function clears *mask* in the format flags.

Definition at line 612 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

**5.377.5.100** `template<typename _CharT, typename _Traits> char_type  
std::basic_ios<_CharT, _Traits>::widen ( char __c ) const  
[inline, inherited]`

Widens characters.

**Parameters**

*c* The character to widen.

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 441 of file `basic_ios.h`.

Referenced by `std::endl()`, `std::basic_ios<char, _Traits>::fill()`, `std::basic_istream<char>::get()`, `std::basic_istream<char>::getline()`, `std::getline()`, and `std::operator>>()`.

#### 5.377.5.101 streamsize std::ios\_base::width ( streamsize \_\_wide ) [inline, inherited]

Changing flags.

**Parameters**

*wide* The new width value.

**Returns**

The previous value of `width()`.

Definition at line 655 of file `ios_base.h`.

#### 5.377.5.102 streamsize std::ios\_base::width ( ) const [inline, inherited]

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 646 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_put<_CharT, _Outputter>::do_put()`, and `std::operator>>()`.

**5.377.5.103** `template<typename _CharT, typename _Traits > basic_ostream<  
_CharT, _Traits > & std::basic_ostream< _CharT, _Traits  
>::write ( const char_type * __s, streamsize __n )  
[inherited]`

Character string insertion.

#### Parameters

- s* The array to insert.
- n* Maximum number of characters to insert.

#### Returns

\*this

Characters are copied from *s* and inserted into the stream until one of the following happens:

- *n* characters are inserted
- inserting into the output sequence fails (in this case, badbit will be set in the stream's error state)

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 185 of file ostream.tcc.

References `std::basic_ostream< _CharT, _Traits >::_M_write()`, and `std::ios_base::badbit`.

**5.377.5.104** `static int std::ios_base::xalloc ( ) throw () [static,  
inherited]`

Access to unique indices.

#### Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

### 5.377.6 Member Data Documentation

#### 5.377.6.1 `template<typename _CharT, typename _Traits> streamsize std::basic_istream<_CharT, _Traits>::_M_gcount` [**protected**, **inherited**]

The number of characters extracted in the previous unformatted function; see [gcount\(\)](#).

Definition at line 81 of file `istream`.

Referenced by `std::basic_istream<char>::gcount()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::readsome()`, `std::basic_istream<_CharT, _Traits>::unget()`, and `std::basic_istream<char>::~~basic_istream()`.

#### 5.377.6.2 `const fmtflags std::ios_base::adjustfield` [**static**, **inherited**]

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Definition at line 312 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

#### 5.377.6.3 `const openmode std::ios_base::app` [**static**, **inherited**]

Seek to end before each write.

Definition at line 366 of file `ios_base.h`.

**5.377.6.4 `const openmode std::ios_base::ate` [`static`, `inherited`]**

Open and seek to end immediately after opening.

Definition at line 369 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

**5.377.6.5 `const iostate std::ios_base::badbit` [`static`, `inherited`]**

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 336 of file `ios_base.h`.

Referenced by `std::basic_ostream< char >::_M_write()`, `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::basic_ios< char, _Traits >::fail()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ios< _CharT, _Traits >::init()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, `std::basic_istream< _CharT, _Traits >::unget()`, `std::basic_ostream< _CharT, _Traits >::write()`, and `std::basic_ostream< _CharT, _Traits >::sentry::~sentry()`.

**5.377.6.6 `const fmtflags std::ios_base::basefield` [`static`, `inherited`]**

A mask of `dec|oct|hex`. Useful for the 2-arg form of `setf`.

Definition at line 315 of file `ios_base.h`.

Referenced by `std::dec()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::hex()`, `std::oct()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

**5.377.6.7 const seekdir std::ios\_base::beg [static, inherited]**

Request a seek relative to the beginning of the stream.

Definition at line 398 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::seekpos().

**5.377.6.8 const openmode std::ios\_base::binary [static, inherited]**

Perform input and output in binary mode (as opposed to text mode). /// This is probably not what you think it is; see /// <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 374 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::showmanyc().

**5.377.6.9 const fmtflags std::ios\_base::boolalpha [static, inherited]**

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 260 of file ios\_base.h.

Referenced by std::boolalpha(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::noboolalpha().

**5.377.6.10 const seekdir std::ios\_base::cur [static, inherited]**

Request a seek relative to the current position within the sequence.

Definition at line 401 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::imbue(), std::basic\_filebuf< \_CharT, \_Traits >::overflow(), std::basic\_filebuf< \_CharT, \_Traits >::pbackfail(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_filebuf< \_CharT, \_Traits >::seekoff(), std::basic\_istream< \_CharT, \_Traits >::tellg(), and std::basic\_ostream< \_CharT, \_Traits >::tellp().



**5.377.6.11 const fmtflags std::ios\_base::dec [static, inherited]**

Converts integer input or generates integer output in decimal base.

Definition at line 263 of file ios\_base.h.

Referenced by std::dec().

**5.377.6.12 const seekdir std::ios\_base::end [static, inherited]**

Request a seek relative to the current end of the sequence.

Definition at line 404 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::open(), and std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff().

**5.377.6.13 const iostate std::ios\_base::eofbit [static, inherited]**

Indicates that an input operation reached the end of an input sequence.

Definition at line 339 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_date(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::eof(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::unget(), and std::ws().

**5.377.6.14 const iostate std::ios\_base::failbit [static, inherited]**

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 344 of file ios\_base.h.

Referenced by std::basic\_fstream< \_CharT, \_Traits >::close(), std::basic\_ofstream< \_CharT, \_Traits >::close(), std::basic\_ifstream< \_CharT, \_Traits >::close(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::fail(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_fstream< \_CharT, \_Traits >::open(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry(), and std::basic\_istream< \_CharT, \_Traits >::sentry::sentry().

#### 5.377.6.15 const fmtflags std::ios\_base::fixed [static, inherited]

Generate floating-point output in fixed-point notation.

Definition at line 266 of file ios\_base.h.

Referenced by std::fixed().

#### 5.377.6.16 const fmtflags std::ios\_base::floatfield [static, inherited]

A mask of scientific|fixed. Useful for the 2-arg form of setf.

Definition at line 318 of file ios\_base.h.

Referenced by std::fixed(), and std::scientific().

#### 5.377.6.17 const iostate std::ios\_base::goodbit [static, inherited]

Indicates all is well.

Definition at line 347 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(),

`std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_ios<_CharT, _Traits>::init()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_ostream<_CharT, _Traits>::put()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::readsome()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, `std::basic_istream<_CharT, _Traits>::sync()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

#### 5.377.6.18 `const fmtflags std::ios_base::hex` [static, inherited]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 269 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, `std::hex()`, and `std::basic_ostream<_CharT, _Traits>::operator<<()`.

#### 5.377.6.19 `const openmode std::ios_base::in` [static, inherited]

Open for input. Default for `ifstream` and `fstream`.

Definition at line 377 of file `ios_base.h`.

Referenced by `std::basic_filebuf<char_type, traits_type>::_M_set_buffer()`, `std::basic_ifstream<_CharT, _Traits>::open()`, `std::basic_filebuf<_CharT, _Traits>::pbackfail()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::showmanyc()`, `std::basic_filebuf<_CharT, _Traits>::showmanyc()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::underflow()`, `std::basic_filebuf<_CharT, _Traits>::underflow()`, and `std::basic_filebuf<_CharT, _Traits>::xsgetn()`.

#### 5.377.6.20 `const fmtflags std::ios_base::internal` [static, inherited]

Adds fill characters at a designated internal point in certain `///` generated output, or identical to `right` if no such point is `///` designated.

Definition at line 274 of file ios\_base.h.

Referenced by std::internal().

**5.377.6.21 const fmtflags std::ios\_base::left [static, inherited]**

Adds fill characters on the right (final positions) of certain /// generated output. (I.e., the thing you print is flush left.).

Definition at line 278 of file ios\_base.h.

Referenced by std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::left().

**5.377.6.22 const fmtflags std::ios\_base::oct [static, inherited]**

Converts integer input or generates integer output in octal base.

Definition at line 281 of file ios\_base.h.

Referenced by std::oct(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

**5.377.6.23 const openmode std::ios\_base::out [static, inherited]**

Open for output. Default for ofstream and fstream.

Definition at line 380 of file ios\_base.h.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::M\_set\_buffer(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::overflow(), std::basic\_filebuf< \_CharT, \_Traits >::overflow(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::pbackfail(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos(), std::basic\_ostream< \_CharT, \_Traits >::tellp(), and std::basic\_filebuf< \_CharT, \_Traits >::xsputn().

**5.377.6.24 const fmtflags std::ios\_base::right [static, inherited]**

Adds fill characters on the left (initial positions) of certain /// generated output. (I.e., the thing you print is flush right.).

Definition at line 285 of file ios\_base.h.

Referenced by std::right().

**5.377.6.25 const fmtflags std::ios\_base::scientific [static, inherited]**

Generates floating-point output in scientific notation.

Definition at line 288 of file ios\_base.h.

Referenced by std::scientific().

**5.377.6.26 const fmtflags std::ios\_base::showbase [static, inherited]**

Generates a prefix indicating the numeric base of generated integer /// output.

Definition at line 292 of file ios\_base.h.

Referenced by std::noshowbase(), and std::showbase().

**5.377.6.27 const fmtflags std::ios\_base::showpoint [static, inherited]**

Generates a decimal-point character unconditionally in generated /// floating-point output.

Definition at line 296 of file ios\_base.h.

Referenced by std::noshowpoint(), and std::showpoint().

**5.377.6.28 const fmtflags std::ios\_base::showpos [static, inherited]**

Generates a + sign in non-negative generated numeric output.

Definition at line 299 of file ios\_base.h.

Referenced by std::noshowpos(), and std::showpos().

**5.377.6.29 const fmtflags std::ios\_base::skipws [static, inherited]**

Skips leading white space before certain input operations.

## 5.378 `std::basic_ifstream<_CharT, _Traits>` Class Template Reference 1723

Definition at line 302 of file `ios_base.h`.

Referenced by `std::noskipws()`, `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, and `std::skipws()`.

### 5.377.6.30 `const openmode std::ios_base::trunc` [static, inherited]

Open for input. Default for `ofstream`.

Definition at line 383 of file `ios_base.h`.

### 5.377.6.31 `const fmtflags std::ios_base::unitbuf` [static, inherited]

Flushes output after each output operation.

Definition at line 305 of file `ios_base.h`.

Referenced by `std::nunitbuf()`, `std::unitbuf()`, and `std::basic_ostream<_CharT, _Traits>::sentry::~sentry()`.

### 5.377.6.32 `const fmtflags std::ios_base::uppercase` [static, inherited]

Replaces certain lowercase letters with their uppercase equivalents /// in generated output.

Definition at line 309 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`, `std::nouppercase()`, and `std::uppercase()`.

The documentation for this class was generated from the following file:

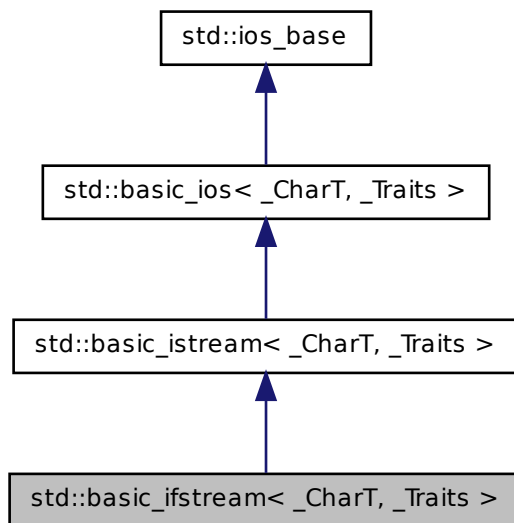
- [fstream](#)

## 5.378 `std::basic_ifstream<_CharT, _Traits>` Class Template Reference

Controlling input for files.

This class supports reading from named files, using the inherited functions from [std::basic\\_istream](#). To control the associated sequence, an instance of [std::basic\\_filebuf](#) is used, which this page refers to as `sb`.

Inheritance diagram for `std::basic_ifstream< _CharT, _Traits >`:



### Public Types

- typedef `ctype< _CharT > __ctype_type`
- typedef `basic_filebuf< char_type, traits_type > __filebuf_type`
- typedef `basic_ios< _CharT, _Traits > __ios_type`
- typedef `basic_istream< char_type, traits_type > __istream_type`
- typedef `num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > __num_get_type`
- typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`
- typedef `_CharT char_type`
- enum `event { erase_event, imbue_event, copyfmt_event }`
- typedef `void(* event_callback)(event, ios_base &, int)`
- typedef `_Ios_Fmtflags fmtflags`
- typedef `traits_type::int_type int_type`
- typedef `int io_state`
- typedef `_Ios_Iostate iostate`
- typedef `traits_type::off_type off_type`

- typedef int **open\_mode**
- typedef `_Ios_Openmode` **openmode**
- typedef `traits_type::pos_type` **pos\_type**
- typedef int **seek\_dir**
- typedef `_Ios_Seekdir` **seekdir**
- typedef `std::streamoff` **streamoff**
- typedef `std::streampos` **streampos**
- typedef `_Traits` **traits\_type**
  
- typedef `num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __-`  
    `num_put_type`

### Public Member Functions

- `basic_ifstream ()`
- `basic_ifstream (const char *__s, ios_base::openmode __mode=ios_base::in)`
- `basic_ifstream (const std::string &__s, ios_base::openmode __mode=ios_base::in)`
- `~basic_ifstream ()`
- `const locale & _M_getloc () const`
- `void _M_setstate (iostate __state)`
- `bool bad () const`
- `void clear (iostate __state=goodbit)`
- `void close ()`
- `basic_ios & copyfmt (const basic_ios &__rhs)`
- `bool eof () const`
- `iostate exceptions () const`
- `void exceptions (iostate __except)`
- `bool fail () const`
- `char_type fill () const`
- `char_type fill (char_type __ch)`
- `fmtflags flags () const`
- `fmtflags flags (fmtflags __fmtfl)`
- `streamsize gcount () const`
- `template<>`  
    `basic_istream< char > & getline (char_type *__s, streamsize __n, char_type`  
    `__delim)`
- `template<>`  
    `basic_istream< wchar_t > & getline (char_type *__s, streamsize __n, char_type`  
    `__delim)`
- `locale getloc () const`
- `bool good () const`



- `template<>`  
`basic_ifstream< wchar_t > & ignore (streamsize __n)`
  - `template<>`  
`basic_ifstream< wchar_t > & ignore (streamsize __n, int_type __delim)`
  - `template<>`  
`basic_ifstream< char > & ignore (streamsize __n)`
  - `template<>`  
`basic_ifstream< char > & ignore (streamsize __n, int_type __delim)`
  - `locale imbue (const locale &__loc)`
  - `bool is_open ()`
  - `bool is_open () const`
  - `long & iword (int __ix)`
  - `char narrow (char_type __c, char __dfault) const`
  - `void open (const char *__s, ios_base::openmode __mode=ios_base::in)`
  - `void open (const std::string &__s, ios_base::openmode __mode=ios_base::in)`
  - `streamsize precision () const`
  - `streamsize precision (streamsize __prec)`
  - `void *& pword (int __ix)`
  - `__filebuf_type * rdbuf () const`
  - `basic_streambuf<_CharT, _Traits> * rdbuf (basic_streambuf<_CharT, _Traits> * __sb)`
  - `iosstate rdstate () const`
  - `void register_callback (event_callback __fn, int __index)`
  - `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
  - `fmtflags setf (fmtflags __fmtfl)`
  - `void setstate (iosstate __state)`
  - `basic_ostream<_CharT, _Traits> * tie () const`
  - `basic_ostream<_CharT, _Traits> * tie (basic_ostream<_CharT, _Traits> * __tistr)`
  - `void unsetf (fmtflags __mask)`
  - `char_type widen (char __c) const`
  - `streamsize width () const`
  - `streamsize width (streamsize __wide)`
- 
- `__istream_type & operator>> (__istream_type &(__pf)(__istream_type &))`
  - `__istream_type & operator>> (__ios_type &(__pf)(__ios_type &))`
  - `__istream_type & operator>> (ios_base &(__pf)(ios_base &))`

### Arithmetic Extractors

*All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ifstream::sentry` with the second argument (`noskipws`) set to false. This has several*

effects, concluding with the setting of a status flag; see the [sentry documentation](#) for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, [`ios\_base::badbit`](#) will be turned on in the stream's error state without causing an [`ios\_base::failure`](#) to be thrown. The original exception will then be rethrown.

- [`\_\_istream\_type & operator>> \(bool &\_\_n\)`](#)
- [`\_\_istream\_type & operator>> \(short &\_\_n\)`](#)
- [`\_\_istream\_type & operator>> \(unsigned short &\_\_n\)`](#)
- [`\_\_istream\_type & operator>> \(int &\_\_n\)`](#)
- [`\_\_istream\_type & operator>> \(unsigned int &\_\_n\)`](#)
- [`\_\_istream\_type & operator>> \(long &\_\_n\)`](#)
- [`\_\_istream\_type & operator>> \(unsigned long &\_\_n\)`](#)
- [`\_\_istream\_type & operator>> \(long long &\_\_n\)`](#)
- [`\_\_istream\_type & operator>> \(unsigned long long &\_\_n\)`](#)
- [`\_\_istream\_type & operator>> \(float &\_\_f\)`](#)
- [`\_\_istream\_type & operator>> \(double &\_\_f\)`](#)
- [`\_\_istream\_type & operator>> \(long double &\_\_f\)`](#)
- [`\_\_istream\_type & operator>> \(void \*&\_\_p\)`](#)
- [`\_\_istream\_type & operator>> \(\_\_streambuf\_type \*\_\_sb\)`](#)

### Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type [`std::basic\_istream::sentry`](#) with the second argument (`noskipws`) set to true. This has several effects, concluding with the setting of a status flag; see the [sentry documentation](#) for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by [`gcount\(\)`](#).

If an exception is thrown during extraction, [`ios\_base::badbit`](#) will be turned on in the stream's error state without causing an [`ios\_base::failure`](#) to be thrown. The original exception will then be rethrown.

- [`int\_type get \(\)`](#)
- [`\_\_istream\_type & get \(char\_type &\_\_c\)`](#)
- [`\_\_istream\_type & get \(char\_type \*\_\_s, streamsize \_\_n, char\_type \_\_delim\)`](#)
- [`\_\_istream\_type & get \(char\_type \*\_\_s, streamsize \_\_n\)`](#)
- [`\_\_istream\_type & get \(\_\_streambuf\_type &\_\_sb, char\_type \_\_delim\)`](#)
- [`\_\_istream\_type & get \(\_\_streambuf\_type &\_\_sb\)`](#)
- [`\_\_istream\_type & getline \(char\_type \*\_\_s, streamsize \_\_n, char\_type \_\_delim\)`](#)
- [`\_\_istream\_type & getline \(char\_type \*\_\_s, streamsize \_\_n\)`](#)
- [`\_\_istream\_type & ignore \(\)`](#)
- [`\_\_istream\_type & ignore \(streamsize \_\_n\)`](#)

- `__istream_type & ignore (streamsize __n, int_type __delim)`
- `int_type peek ()`
- `__istream_type & read (char_type *__s, streamsize __n)`
- `streamsize readsome (char_type *__s, streamsize __n)`
- `__istream_type & putback (char_type __c)`
- `__istream_type & unget ()`
- `int sync ()`
- `pos_type tellg ()`
- `__istream_type & seekg (pos_type)`
- `__istream_type & seekg (off_type, ios_base::seekdir)`
  
- `operator void * () const`
- `bool operator! () const`

#### Static Public Member Functions

- static `bool sync_with_stdio (bool __sync=true)`
- static `int xalloc () throw ()`

#### Static Public Attributes

- static const `fmtflags adjustfield`
- static const `openmode app`
- static const `openmode ate`
- static const `iosstate badbit`
- static const `fmtflags basefield`
- static const `seekdir beg`
- static const `openmode binary`
- static const `fmtflags boolalpha`
- static const `seekdir cur`
- static const `fmtflags dec`
- static const `seekdir end`
- static const `iosstate eofbit`
- static const `iosstate failbit`
- static const `fmtflags fixed`
- static const `fmtflags floatfield`
- static const `iosstate goodbit`
- static const `fmtflags hex`
- static const `openmode in`
- static const `fmtflags internal`
- static const `fmtflags left`
- static const `fmtflags oct`
- static const `openmode out`
- static const `fmtflags right`

- static const [fmtflags scientific](#)
- static const [fmtflags showbase](#)
- static const [fmtflags showpoint](#)
- static const [fmtflags showpos](#)
- static const [fmtflags skipws](#)
- static const [openmode trunc](#)
- static const [fmtflags unitbuf](#)
- static const [fmtflags uppercase](#)

### Protected Types

- enum { `_S_local_word_size` }

### Protected Member Functions

- void `_M_cache_locale` (const [locale](#) &\_\_loc)
- void `_M_call_callbacks` ([event](#) \_\_ev) throw ()
- void `_M_dispose_callbacks` (void) throw ()
- template<typename \_ValueT >  
  [\\_\\_istream\\_type](#) & `_M_extract` (\_ValueT &\_\_v)
- `_Words` & `_M_grow_words` (int \_\_index, bool \_\_iword)
- void `_M_init` () throw ()
- void `init` ([basic\\_streambuf](#)< \_CharT, \_Traits > \*\_\_sb)

### Protected Attributes

- `_Callback_list` \* `_M_callbacks`
- const [\\_\\_ctype\\_type](#) \* `_M_ctype`
- [iostate](#) `_M_exception`
- [char\\_type](#) `_M_fill`
- bool `_M_fill_init`
- [fmtflags](#) `_M_flags`
- [streamsize](#) `_M_gcount`
- [locale](#) `_M_ios_locale`
- `_Words` `_M_local_word` [`_S_local_word_size`]
- const [\\_\\_num\\_get\\_type](#) \* `_M_num_get`
- const [\\_\\_num\\_put\\_type](#) \* `_M_num_put`
- [streamsize](#) `_M_precision`
- [basic\\_streambuf](#)< \_CharT, \_Traits > \* `_M_streambuf`
- [iostate](#) `_M_streambuf_state`
- [basic\\_ostream](#)< \_CharT, \_Traits > \* `_M_tie`
- [streamsize](#) `_M_width`

- `_Words * _M_word`
- `int _M_word_size`
- `_Words _M_word_zero`

## Friends

- class `sentry`

### 5.378.1 Detailed Description

`template<typename _CharT, typename _Traits> class std::basic_ifstream< _CharT, _Traits >`

Controlling input for files.

This class supports reading from named files, using the inherited functions from [std::basic\\_istream](#). To control the associated sequence, an instance of [std::basic\\_filebuf](#) is used, which this page refers to as `sb`.

Definition at line 420 of file `fstream`.

### 5.378.2 Member Typedef Documentation

**5.378.2.1** `template<typename _CharT, typename _Traits> typedef  
    ctype<_CharT> std::basic_istream< _CharT, _Traits  
    >::__ctype_type [inherited]`

These are non-standard types.

Reimplemented from [std::basic\\_ios< \\_CharT, \\_Traits >](#).

Definition at line 73 of file `istream`.

**5.378.2.2** `template<typename _CharT, typename _Traits> typedef  
    num_get<_CharT, istreambuf_iterator<_CharT, _Traits>  
    > std::basic_istream< _CharT, _Traits >::__num_get_type  
    [inherited]`

These are non-standard types.

Reimplemented from [std::basic\\_ios< \\_CharT, \\_Traits >](#).

Definition at line 72 of file `istream`.

**5.378.2.3** `template<typename _CharT, typename _Traits> typedef  
num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>  
> std::basic_ios< _CharT, _Traits >::__num_put_type  
[inherited]`

These are non-standard types.

Reimplemented in [std::basic\\_ostream< \\_CharT, \\_Traits >](#), [std::basic\\_ostream< char, \\_Traits >](#), and [std::basic\\_ostream< char >](#).

Definition at line 86 of file `basic_ios.h`.

**5.378.2.4** `template<typename _CharT, typename _Traits > typedef _CharT  
std::basic_ifstream< _CharT, _Traits >::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_istream< \\_CharT, \\_Traits >](#).

Definition at line 424 of file `fstream`.

**5.378.2.5** `typedef void(* std::ios_base::event_callback)(event, ios_base &, int)  
[inherited]`

The type of an event callback function.

#### Parameters

*event* One of the members of the event enum.

*ios\_base* Reference to the [ios\\_base](#) object.

*int* The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several [ios\\_base](#) and [basic\\_ios](#) functions, specifically [imbue\(\)](#), [copyfmt\(\)](#), and [~ios\(\)](#).

Definition at line 438 of file `ios_base.h`.

**5.378.2.6** `typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]`

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 257 of file `ios_base.h`.

**5.378.2.7** `template<typename _CharT, typename _Traits> typedef traits_type::int_type std::basic_ifstream<_CharT, _Traits>::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_istream<\\_CharT, \\_Traits>](#).

Definition at line 426 of file `fstream`.

**5.378.2.8 `typedef _Ios_Iostate std::ios_base::iostate` [inherited]**

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 332 of file `ios_base.h`.

**5.378.2.9 `template<typename _CharT, typename _Traits> typedef traits_type::off_type std::basic_ifstream<_CharT, _Traits>::off_type`**

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_istream<\\_CharT, \\_Traits>](#).

Definition at line 428 of file `fstream`.

**5.378.2.10 `typedef _Ios_Openmode std::ios_base::openmode` [inherited]**

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`



## 5.378 `std::basic_ifstream< _CharT, _Traits >` Class Template Reference 1734

---

- out
- trunc

Definition at line 363 of file `ios_base.h`.

**5.378.2.11** `template<typename _CharT, typename _Traits > typedef traits_type::pos_type std::basic_ifstream< _CharT, _Traits >::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_istream< _CharT, _Traits >`.

Definition at line 427 of file `fstream`.

**5.378.2.12** `typedef _Ios_Seekdir std::ios_base::seekdir [inherited]`

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- beg
- cur, equivalent to `SEEK_CUR` in the C standard library.
- end, equivalent to `SEEK_END` in the C standard library.

Definition at line 395 of file `ios_base.h`.

**5.378.2.13** `template<typename _CharT, typename _Traits > typedef _Traits std::basic_ifstream< _CharT, _Traits >::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_istream< _CharT, _Traits >`.

Definition at line 425 of file `fstream`.

### 5.378.3 Member Enumeration Documentation

#### 5.378.3.1 enum std::ios\_base::event [inherited]

The set of events that may be passed to an event callback.

erase\_event is used during ~ios() and copyfmt(). imbue\_event is used during [imbue\(\)](#). copyfmt\_event is used during copyfmt().

Definition at line 421 of file ios\_base.h.

### 5.378.4 Constructor & Destructor Documentation

#### 5.378.4.1 template<typename \_CharT , typename \_Traits > std::basic\_ifstream< \_CharT, \_Traits >::basic\_ifstream ( ) [inline]

Default constructor.

Initializes *sb* using its default constructor, and passes *&sb* to the base class initializer. Does not open any files (you haven't given it a filename to open).

Definition at line 446 of file fstream.

References `std::basic_ios< _CharT, _Traits >::init()`.

#### 5.378.4.2 template<typename \_CharT , typename \_Traits > std::basic\_ifstream< \_CharT, \_Traits >::basic\_ifstream ( const char \* \_\_s, ios\_base::openmode \_\_mode = ios\_base::in ) [inline, explicit]

Create an input file stream.

#### Parameters

*s* Null terminated string specifying the filename.

*mode* Open file in specified mode (see [std::ios\\_base](#)).

[ios\\_base::in](#) is automatically included in *mode*.

Tip: When using `std::string` to hold the filename, you must use `.c_str()` before passing it to this constructor.

Definition at line 460 of file `fstream`.

References `std::basic_ios< _CharT, _Traits >::init()`, and `std::basic_ifstream< _CharT, _Traits >::open()`.

**5.378.4.3** `template<typename _CharT , typename _Traits >`  
`std::basic_ifstream< _CharT, _Traits >::basic_ifstream ( const`  
`std::string & __s, ios_base::openmode __mode = ios_base::in )`  
`[inline, explicit]`

Create an input file stream.

#### Parameters

*s* `std::string` specifying the filename.

*mode* Open file in specified mode (see [std::ios\\_base](#)).

`ios_base::in` is automatically included in *mode*.

Definition at line 476 of file `fstream`.

References `std::basic_ios< _CharT, _Traits >::init()`, and `std::basic_ifstream< _CharT, _Traits >::open()`.

**5.378.4.4** `template<typename _CharT , typename _Traits >`  
`std::basic_ifstream< _CharT, _Traits >::~~basic_ifstream ( )`  
`[inline]`

The destructor does nothing.

The file is closed by the `filebuf` object, not the formatting stream.

Definition at line 491 of file `fstream`.

#### 5.378.5 Member Function Documentation

**5.378.5.1** `const locale& std::ios_base::_M_getloc ( ) const [inline,`  
`inherited]`

Locale access.

**Returns**

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 708 of file `ios_base.h`.

Referenced by `std::money_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_time()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::time_put< _CharT, _OutIter >::do_put()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::time_put< _CharT, _OutIter >::put()`.

### 5.378.5.2 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::bad ( ) const [inline, inherited]`

Fast error checking.

**Returns**

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 203 of file `basic_ios.h`.

### 5.378.5.3 `template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::clear ( iostate __state = goodbit ) [inherited]`

[Re]sets the error state.

**Parameters**

*state* The new state flag(s) to set.

See [std::ios\\_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

Definition at line 42 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::exceptions()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::rdstate()`.

Referenced by `std::basic_ios< char, _Traits >::exceptions()`, `std::basic_fstream< _CharT, _Traits >::open()`, `std::basic_ofstream< _CharT, _Traits >::open()`, `std::basic_ifstream< _CharT, _Traits >::open()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ios< char, _Traits >::setstate()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

#### 5.378.5.4 `template<typename _CharT, typename _Traits> void std::basic_ifstream< _CharT, _Traits >::close ( ) [inline]`

Close the file.

Calls `std::basic_filebuf::close()`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 569 of file `fstream`.

References `std::basic_filebuf< _CharT, _Traits >::close()`, `std::ios_base::failbit`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

#### 5.378.5.5 `template<typename _CharT, typename _Traits> basic_ios< _CharT, _Traits> & std::basic_ios< _CharT, _Traits >::copyfmt ( const basic_ios< _CharT, _Traits > & __rhs ) [inherited]`

Copies fields of `__rhs` into this.

#### Parameters

`__rhs` The source values for the copies.

#### Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy `exceptions()`.

Definition at line 64 of file `basic_ios.tcc`.

References `std::basic_ios< _CharT, _Traits >::exceptions()`, `std::basic_ios< _CharT, _Traits >::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios< _CharT, _Traits >::tie()`, and `std::ios_base::width()`.

**5.378.5.6** `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits >::eof( ) const [inline, inherited]`

Fast error checking.

#### Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 182 of file basic\_ios.h.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

**5.378.5.7** `template<typename _CharT, typename _Traits> iostate std::basic_ios< _CharT, _Traits >::exceptions( ) const [inline, inherited]`

Throwing exceptions on errors.

#### Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of [exceptions\(iostate\)](#) for the meaning of the return value.

Definition at line 214 of file basic\_ios.h.

Referenced by `std::basic_ios< _CharT, _Traits >::clear()`, and `std::basic_ios< _CharT, _Traits >::copyfmt()`.

**5.378.5.8** `template<typename _CharT, typename _Traits> void std::basic_ios<_CharT, _Traits >::exceptions( iostate __except ) [inline, inherited]`

Throwing exceptions on errors.

**Parameters**

*except* The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
 std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

 std::ifstream f ("/etc/motd");

 std::cerr << "Setting badbit\n";
 f.setstate (std::ios_base::badbit);

 std::cerr << "Setting exception mask\n";
 f.exceptions (std::ios_base::badbit);
}
```

Definition at line 249 of file `basic_ios.h`.

### 5.378.5.9 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::fail( ) const [inline, inherited]`

Fast error checking.

**Returns**

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Definition at line 193 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::operator void *()`, `std::basic_ios< char, _Traits >::operator!()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::regex_traits< _Ch_type >::value()`.

**5.378.5.10** `template<typename _CharT, typename _Traits> char_type  
std::basic_ios< _CharT, _Traits >::fill ( char_type __ch )  
[inline, inherited]`

Sets a new *empty* character.

#### Parameters

*ch* The new character.

#### Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 382 of file `basic_ios.h`.

**5.378.5.11** `template<typename _CharT, typename _Traits> char_type  
std::basic_ios< _CharT, _Traits >::fill ( ) const [inline,  
inherited]`

Retrieves the *empty* character.

#### Returns

The current fill character.

It defaults to a space ( ' ') in the current locale.

Definition at line 362 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, and `std::basic_ios< char, _Traits >::fill()`.

**5.378.5.12** `fmtflags std::ios_base::flags ( fmtflags __fmtfl ) [inline,  
inherited]`

Setting new format flags all at once.



### Parameters

*fmtfl* The new flags to set.

### Returns

The previous format control flags.

This function overwrites all the format flags with *fmtfl*.

Definition at line 564 of file `ios_base.h`.

#### 5.378.5.13 `fmtflags std::ios_base::flags ( ) const [inline, inherited]`

Access to format flags.

### Returns

The format control flags for both input and output.

Definition at line 553 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, `std::operator>>()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

#### 5.378.5.14 `template<typename _CharT, typename _Traits> streamsize std::basic_istream< _CharT, _Traits >::gcount ( ) const [inline, inherited]`

Character counting.

### Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 251 of file `istream`.

#### 5.378.5.15 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::get ( void ) [inherited]`

Simple extraction.

### Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 238 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.378.5.16** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get ( char_type & __c ) [inherited]`

Simple extraction.

### Parameters

*c* The character in which to store data.

### Returns

`*this`

Tries to extract a character and store it in *c*. If none are available, sets failbit and returns `traits::eof()`.

### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 274 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.378.5.17** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get ( char_type * __s, streamsize __n, char_type __delim ) [inherited]`

Simple multiple-character extraction.

#### Parameters

- s* Pointer to an array.
- n* Maximum number of characters to store in *s*.
- delim* A "stop" character.

#### Returns

\*this

Characters are extracted and stored into *s* until one of the following happens:

- *n*−1 characters are stored
- the input sequence reaches EOF
- the next character equals *delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 311 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

**5.378.5.18** `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits> & std::basic_istream< _CharT, _Traits>::get ( __streambuf_type & __sb, char_type __delim ) [inherited]`

Extraction into another streambuf.

#### Parameters

- sb* A streambuf in which to store data.

*delim* A "stop" character.

### Returns

\*this

Characters are extracted and inserted into *sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 358 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits >::\_M\_gcount, std::ios\_base::badbit, std::basic\_ios< \_CharT, \_Traits >::eof(), std::ios\_base::eofbit, std::ios\_base::failbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), std::basic\_ios< \_CharT, \_Traits >::setstate(), std::basic\_streambuf< \_CharT, \_Traits >::sgetc(), std::basic\_streambuf< \_CharT, \_Traits >::snextc(), and std::basic\_streambuf< \_CharT, \_Traits >::sputc().

**5.378.5.19** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::get ( char_type * __s, streamsize __n ) [inline, inherited]`

Simple multiple-character extraction.

### Parameters

*s* Pointer to an array.

*n* Maximum number of characters to store in *s*.

### Returns

\*this

Returns `get(s,n,widen('\n'))`.

Definition at line 335 of file istream.

**5.378.5.20** `template<typename _CharT, typename _Traits> __istream_type& std::basic_ifstream<_CharT, _Traits>::get ( __streambuf_type & __sb ) [inline, inherited]`

Extraction into another streambuf.

#### Parameters

*sb* A streambuf in which to store data.

#### Returns

\*this

Returns `get(sb, widen('\n'))`.

Definition at line 368 of file `istream`.

**5.378.5.21** `template<typename _CharT, typename _Traits> __istream_type& std::basic_ifstream<_CharT, _Traits>::getline ( char_type * __s, streamsize __n ) [inline, inherited]`

String extraction.

#### Parameters

*s* A character array in which to store the data.

*n* Maximum number of characters to extract.

#### Returns

\*this

Returns `getline(s, n, widen('\n'))`.

Definition at line 408 of file `istream`.

Referenced by `std::basic_ifstream<char>::getline()`.

**5.378.5.22** `template<typename _CharT, typename _Traits> basic_ifstream<_CharT, _Traits> & std::basic_ifstream<_CharT, _Traits>::getline ( char_type * __s, streamsize __n, char_type __delim ) [inherited]`

String extraction.

#### Parameters

- s* A character array in which to store the data.
- n* Maximum number of characters to extract.
- delim* A "stop" character.

#### Returns

`*this`

Extracts and stores characters into *s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals *delim*, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. *n*−1 characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 402 of file `istream.tcc`.

References `std::basic_ifstream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_streambuf< _CharT, _Traits >::sbumpc()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

#### 5.378.5.23 `std::ios_base::getloc()` `const` `[inline, inherited]`

Locale access.

#### Returns

A copy of the current locale.

## 5.378 `std::basic_ifstream<_CharT, _Traits>` Class Template Reference 1748

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 697 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::money_put<_CharT, _Outputter>::do_put()`, `std::basic_ios<_CharT, _Traits>::imbue()`, `std::operator>>()`, and `std::ws()`.

**5.378.5.24** `template<typename _CharT, typename _Traits> bool  
std::basic_ios<_CharT, _Traits>::good( ) const [inline,  
inherited]`

Fast error checking.

### Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 172 of file `basic_ios.h`.

Referenced by `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

**5.378.5.25** `template<typename _CharT, typename _Traits> basic_istream<  
_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore  
( void ) [inherited]`

Discarding characters.

### Parameters

*n* Number of characters to discard.

*delim* A "stop" character.

### Returns

`*this`

Extracts characters and throws them away until one of the following happens:

- if `n != std::numeric_limits<int>::max()`, `n` characters are extracted

- the input sequence reaches end-of-file
- the next character equals *delim* (in this case, the character is extracted); note that this condition will never occur if *delim* equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 462 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_streambuf< _CharT, _Traits >::sbumpc()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.378.5.26** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore ( streamsize __n ) [inherited]`

Simple extraction.

#### Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 495 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

**5.378.5.27** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore ( streamsize __n, int_type __delim ) [inherited]`

Simple extraction.

#### Returns

A character, or `eof()`.



## 5.378 std::basic\_ifstream< \_CharT, \_Traits > Class Template Reference 1750

---

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 557 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits >::\_M\_gcount, std::ios\_base::badbit, std::basic\_ios< \_CharT, \_Traits >::eof(), std::ios\_base::eofbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), std::basic\_streambuf< \_CharT, \_Traits >::sbumpc(), std::basic\_ios< \_CharT, \_Traits >::setstate(), std::basic\_streambuf< \_CharT, \_Traits >::sgetc(), and std::basic\_streambuf< \_CharT, \_Traits >::snextc().

### 5.378.5.28 template<typename \_CharT, typename \_Traits > locale std::basic\_ios< \_CharT, \_Traits >::imbue ( const locale & \_\_loc ) [inherited]

Moves to a new locale.

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls ios\_base::imbue(loc), and if a stream buffer is associated with this stream, calls that buffer's pubimbue(loc).

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.ht>

Reimplemented from std::ios\_base.

Definition at line 115 of file basic\_ios.tcc.

References std::ios\_base::getloc(), and std::basic\_ios< \_CharT, \_Traits >::rdbuf().

Referenced by std::operator<<().

### 5.378.5.29 template<typename \_CharT, typename \_Traits> void std::basic\_ios< \_CharT, \_Traits >::init ( basic\_streambuf< \_CharT, \_Traits > \* \_\_sb ) [protected, inherited]

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 127 of file basic\_ios.tcc.

References std::ios\_base::badbit, and std::ios\_base::goodbit.

Referenced by `std::basic_fstream< _CharT, _Traits >::basic_fstream()`, `std::basic_ifstream< _CharT, _Traits >::basic_ifstream()`, `std::basic_ios< char, _Traits >::basic_ios()`, `std::basic_istream< char >::basic_istream()`, `std::basic_istreamstream< _CharT, _Traits, _Alloc >::basic_istreamstream()`, `std::basic_ofstream< _CharT, _Traits >::basic_ofstream()`, `std::basic_ostream< char >::basic_ostream()`, `std::basic_ostringstream< _CharT, _Traits, _Alloc >::basic_ostringstream()`, and `std::basic_stringstream< _CharT, _Traits, _Alloc >::basic_stringstream()`.

**5.378.5.30** `template<typename _CharT, typename _Traits> bool  
std::basic_ifstream< _CharT, _Traits >::is_open ( ) [inline]`

Wrapper to test for an open file.

#### Returns

`rdbuf()` -> `is_open()`

Definition at line 510 of file `fstream`.

References `std::basic_filebuf< _CharT, _Traits >::is_open()`.

**5.378.5.31** `long& std::ios_base::iword ( int __ix ) [inline, inherited]`

Access to integer array.

#### Parameters

`__ix` Index into the array.

#### Returns

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 743 of file `ios_base.h`.

**5.378.5.32** `template<typename _CharT, typename _Traits> char  
std::basic_ios<_CharT, _Traits>::narrow ( char_type __c, char  
__dfault ) const [inline, inherited]`

Squeezes characters.

#### Parameters

- c* The character to narrow.
- dfault* The character to narrow.

#### Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, dfault)
```

Additional 110n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.ht>

Definition at line 422 of file `basic_ios.h`.

**5.378.5.33** `template<typename _CharT, typename _Traits> void  
std::basic_ifstream<_CharT, _Traits>::open ( const std::string &  
__s, ios_base::openmode __mode = ios_base::in ) [inline]`

Opens an external file.

#### Parameters

- s* The name of the file.
- mode* The open mode flags.

Calls `std::basic_filebuf::open(s, mode|in)`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 551 of file `fstream`.

References `std::basic_ios<_CharT, _Traits>::clear()`, `std::ios_base::failbit`, `std::ios_base::in`, `std::basic_filebuf<_CharT, _Traits>::open()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.378.5.34** `template<typename _CharT, typename _Traits> void  
std::basic_ifstream<_CharT, _Traits>::open ( const char * __s,  
ios_base::openmode __mode = ios_base::in ) [inline]`

Opens an external file.

#### Parameters

- s* The name of the file.
- mode* The open mode flags.

Calls `std::basic_filebuf::open(s,mode|in)`. If that function fails, `failbit` is set in the stream's error state.

Tip: When using `std::string` to hold the filename, you must use `.c_str()` before passing it to this constructor.

Definition at line 531 of file `fstream`.

References `std::basic_ios<_CharT, _Traits>::clear()`, `std::ios_base::failbit`, `std::ios_base::in`, `std::basic_filebuf<_CharT, _Traits>::open()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

Referenced by `std::basic_ifstream<_CharT, _Traits>::basic_ifstream()`.

**5.378.5.35** `template<typename _CharT, typename _Traits> std::basic_ios<  
_CharT, _Traits>::operator void * ( ) const [inline,  
inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 113 of file `basic_ios.h`.

**5.378.5.36** `template<typename _CharT, typename _Traits> bool  
std::basic_ios<_CharT, _Traits>::operator! ( ) const  
[inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 117 of file basic\_ios.h.

**5.378.5.37** `template<typename _CharT, typename _Traits> __istream_type&  
std::basic_ifstream< _CharT, _Traits >::operator>> ( __ios_type  
&(*)(__ios_type &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iosmanip` header.

Definition at line 126 of file `istream`.

**5.378.5.38** `template<typename _CharT, typename _Traits> __istream_type&  
std::basic_ifstream< _CharT, _Traits >::operator>> ( unsigned  
long & __n ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

\*`this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 191 of file `istream`.

**5.378.5.39** `template<typename _CharT, typename _Traits> __istream_type&  
std::basic_ifstream< _CharT, _Traits >::operator>> ( long long &  
__n ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 196 of file `istream`.

**5.378.5.40** `template<typename _CharT, typename _Traits> __istream_type& std::basic_ifstream<_CharT, _Traits>::operator>> ( ios_base &(*) (ios_base &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iomanip` header.

Definition at line 133 of file `istream`.

**5.378.5.41** `template<typename _CharT, typename _Traits> __istream_type& std::basic_ifstream<_CharT, _Traits>::operator>> ( float & __f ) [inline, inherited]`

Basic arithmetic extractors.

**Parameters**

`A` variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 205 of file `istream`.

**5.378.5.42** `template<typename _CharT, typename _Traits> __istream_type& std::basic_ifstream<_CharT, _Traits>::operator>> ( double & __f ) [inline, inherited]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 209 of file `istream`.

**5.378.5.43** `template<typename _CharT, typename _Traits> __istream_type& std::basic_ifstream<_CharT, _Traits>::operator>> ( bool & __n ) [inline, inherited]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 169 of file `istream`.

**5.378.5.44** `template<typename _CharT, typename _Traits> __istream_type& std::basic_ifstream<_CharT, _Traits>::operator>> ( long double & __f ) [inline, inherited]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 213 of file `istream`.

**5.378.5.45** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> ( short & __n ) [inherited]`

Basic arithmetic extractors.

#### Parameters

`A` variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 116 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get< _CharT, _InIter >::get()`, `std::ios_base::goodbit`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.378.5.46** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> ( void *& __p ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

`A` variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 217 of file `istream`.



**5.378.5.47** `template<typename _CharT, typename _Traits > basic_ifstream< _CharT, _Traits > & std::basic_ifstream< _CharT, _Traits >::operator>> ( __streambuf_type * __sb ) [inherited]`

Extracting into another streambuf.

#### Parameters

*sb* A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the *sb* streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 206 of file istream.tcc.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.378.5.48** `template<typename _CharT, typename _Traits> __istream_type& std::basic_ifstream< _CharT, _Traits >::operator>> ( unsigned short & __n ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

\*this if successful

These functions use the stream's current locale (specifically, the [num\\_get](#) facet) to parse the input data.

Definition at line 176 of file istream.

```
5.378.5.49 template<typename _CharT, typename _Traits> __istream_type&
std::basic_ifstream< _CharT, _Traits >::operator>> (unsigned
long long & __n) [inline, inherited]
```

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

\**this* if successful

These functions use the stream's current locale (specifically, the [num\\_get](#) facet) to parse the input data.

Definition at line 200 of file istream.

```
5.378.5.50 template<typename _CharT, typename _Traits > basic_ifstream<
_CharT, _Traits > & std::basic_ifstream< _CharT, _Traits
>::operator>> (int & __n) [inherited]
```

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

\**this* if successful

These functions use the stream's current locale (specifically, the [num\\_get](#) facet) to parse the input data.

Definition at line 161 of file istream.tcc.

References [std::ios\\_base::badbit](#), [std::ios\\_base::failbit](#), [std::num\\_get< \\_CharT, \\_InIter >::get\(\)](#), [std::ios\\_base::goodbit](#), and [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#).

**5.378.5.51** `template<typename _CharT, typename _Traits> __istream_type&  
std::basic_ifstream<_CharT, _Traits>::operator>> ( unsigned int  
& __n ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 183 of file `istream`.

**5.378.5.52** `template<typename _CharT, typename _Traits> __istream_type&  
std::basic_ifstream<_CharT, _Traits>::operator>> (   
__istream_type &(*)(__istream_type &) __pf ) [inline,  
inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 122 of file `istream`.

**5.378.5.53** `template<typename _CharT, typename _Traits> __istream_type&  
std::basic_ifstream<_CharT, _Traits>::operator>> ( long & __n  
) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

*\*this* if successful

## 5.378 `std::basic_istream< _CharT, _Traits >` Class Template Reference 1761

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 187 of file `istream`.

**5.378.5.54** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek ( void ) [inherited]`

Looking ahead in the stream.

### Returns

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 622 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.378.5.55** `streamsize std::ios_base::precision ( streamsize __prec ) [inline, inherited]`

Changing flags.

### Parameters

*prec* The new precision value.

### Returns

The previous value of `precision()`.

Definition at line 632 of file `ios_base.h`.

**5.378.5.56** `streamsize std::ios_base::precision ( ) const [inline, inherited]`

Flags access.

### Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 623 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, and `std::operator<<()`.

**5.378.5.57** `template<typename _CharT, typename _Traits > basic_istream<  
_CharT, _Traits > & std::basic_istream< _CharT, _Traits  
>::putback( char_type __c ) [inherited]`

Unextracting a single character.

### Parameters

*c* The character to push back into the input stream.

### Returns

`*this`

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

### Note

Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 713 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::rdstate()`, `std::basic_ios< _CharT, _Traits >::setstate()`, and `std::basic_streambuf< _CharT, _Traits >::sputbackc()`.

Referenced by `std::operator>>()`.

**5.378.5.58** void\*& std::ios\_base::pword ( int \_\_ix ) [inline, inherited]

Access to void pointer array.

#### Parameters

\_\_ix Index into the array.

#### Returns

A reference to a void\* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 764 of file ios\_base.h.

**5.378.5.59** template<typename \_CharT, typename \_Traits > \_\_filebuf\_type\* std::basic\_ifstream< \_CharT, \_Traits >::rdbuf ( ) const [inline]

Accessing the underlying buffer.

#### Returns

The current [basic\\_filebuf](#) buffer.

This hides both signatures of [std::basic\\_ios::rdbuf\(\)](#).

Reimplemented from [std::basic\\_ios< \\_CharT, \\_Traits >](#).

Definition at line 502 of file fstream.

**5.378.5.60** template<typename \_CharT, typename \_Traits> basic\_streambuf< \_CharT, \_Traits > \* std::basic\_ios< \_CharT, \_Traits >::rdbuf ( basic\_streambuf< \_CharT, \_Traits > \* \_\_sb ) [inherited]

Changing the underlying buffer.

**Parameters**

*sb* The new stream buffer.

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream foo; // or some other derived type
std::streambuf* p =;

foo.ios::rdbuf(p); // ios == basic_ios<char>
```

Definition at line 54 of file `basic_ios.tcc`.

References `std::basic_ios< _CharT, _Traits >::clear()`.

**5.378.5.61** `template<typename _CharT, typename _Traits> iostate  
std::basic_ios< _CharT, _Traits >::rdstate ( ) const [inline,  
inherited]`

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 129 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::basic_ios< char, _Traits >::eof()`, `std::basic_ios< char, _Traits >::fail()`, `std::basic_ios< char, _Traits >::good()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ios< char, _Traits >::setstate()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

**5.378.5.62** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read ( char_type * __s, streamsize __n ) [inherited]`

Extraction without delimiters.

#### Parameters

- s* A character array.
- n* Maximum number of characters to store.

#### Returns

\*this

If the stream state is `good()`, extracts characters and stores them into *s* until one of the following happens:

- n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 652 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.378.5.63** `template<typename _CharT, typename _Traits > streamsize std::basic_istream< _CharT, _Traits >::readsome ( char_type * __s, streamsize __n ) [inherited]`

Extraction until the buffer is exhausted, but no more.

#### Parameters

- s* A character array.
- n* Maximum number of characters to store.



**Returns**

The number of characters extracted.

Extracts characters and stores them into *s* depending on the number of characters remaining in the streambuf's buffer, `rddbuf() -> in_avail()`, called *A* here:

- if *A* == -1, sets eofbit and extracts no characters
- if *A* == 0, extracts no characters
- if *A* > 0, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 681 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::min()`, `std::basic_ios< _CharT, _Traits >::rddbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.378.5.64 void std::ios\_base::register\_callback ( event\_callback \_\_fn, int \_\_index ) [inherited]**

Add the callback `__fn` with parameter `__index`.

**Parameters**

`__fn` The function to add.

`__index` The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**5.378.5.65 template<typename \_CharT, typename \_Traits > basic\_istream< \_CharT, \_Traits > & std::basic\_istream< \_CharT, \_Traits >::seekg ( pos\_type \_\_pos ) [inherited]**

Changing the current read position.

**Parameters**

*pos* A file position object.

**Returns**

\*this

If `fail()` is not true, calls `rdbuf() -> pubseekpos(pos)`. If that function fails, sets failbit.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 847 of file istream.tcc.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::ios_base::eofbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::rdstate()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.378.5.66** `template<typename _CharT, typename _Traits > basic_ifstream< _CharT, _Traits > & std::basic_ifstream< _CharT, _Traits >::seekg( off_type __off, ios_base::seekdir __dir ) [inherited]`

Changing the current read position.

**Parameters**

*off* A file offset object.

*dir* The direction in which to seek.

**Returns**

\*this

If `fail()` is not true, calls `rdbuf() -> pubseekoff(off, dir)`. If that function fails, sets failbit.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 886 of file istream.tcc.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::ios_base::eofbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`,

std::ios\_base::goodbit, std::ios\_base::in, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), std::basic\_ios< \_CharT, \_Traits >::rdstate(), and std::basic\_ios< \_CharT, \_Traits >::setstate().

#### 5.378.5.67 fmtflags std::ios\_base::setf ( fmtflags \_\_*fmtfl* ) [inline, inherited]

Setting new format flags.

##### Parameters

*fmtfl* Additional flags to set.

##### Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 580 of file ios\_base.h.

Referenced by std::dec(), std::fixed(), std::hex(), std::left(), std::oct(), std::right(), std::scientific(), std::showbase(), std::showpoint(), std::showpos(), std::skipws(), std::unitbuf(), and std::uppercase().

#### 5.378.5.68 fmtflags std::ios\_base::setf ( fmtflags \_\_*fmtfl*, fmtflags \_\_*mask* ) [inline, inherited]

Setting new format flags.

##### Parameters

*fmtfl* Additional flags to set.

*mask* The flags mask for *fmtfl*.

##### Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is [ios\\_base::adjustfield](#).

Definition at line 597 of file ios\_base.h.

**5.378.5.69** `template<typename _CharT, typename _Traits> void  
std::basic_ios< _CharT, _Traits >::setstate ( iostate __state )  
[inline, inherited]`

Sets additional flags in the error state.

#### Parameters

*state* The additional state flag(s) to set.

See [std::ios\\_base::iostate](#) for the possible bit values.

Definition at line 149 of file `basic_ios.h`.

Referenced by `std::basic_ostream< char >::_M_write()`, `std::basic_fstream< _CharT, _Traits >::close()`, `std::basic_ofstream< _CharT, _Traits >::close()`, `std::basic_ifstream< _CharT, _Traits >::close()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_fstream< _CharT, _Traits >::open()`, `std::basic_ofstream< _CharT, _Traits >::open()`, `std::basic_ifstream< _CharT, _Traits >::open()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

**5.378.5.70** `template<typename _CharT, typename _Traits > int  
std::basic_istream< _CharT, _Traits >::sync ( void )  
[inherited]`

Synchronizing the stream buffer.

#### Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf() -> pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

#### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 783 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_streambuf< _CharT, _Traits >::pubsync()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.378.5.71** `static bool std::ios_base::sync_with_stdio ( bool __sync = true )`  
[static, inherited]

Interaction with the standard C I/O objects.

#### Parameters

*sync* Whether to synchronize or not.

#### Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

**5.378.5.72** `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits>::pos_type std::basic_istream< _CharT, _Traits>::tellg ( void )` [inherited]

Getting the current read position.

#### Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, in)`.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 819 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::in`, and `std::basic_ios< _CharT, _Traits >::rdbuf()`.

**5.378.5.73** `template<typename _CharT, typename _Traits>  
basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits  
>::tie( ) const [inline, inherited]`

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 287 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

**5.378.5.74** `template<typename _CharT, typename _Traits>  
basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits  
>::tie( basic_ostream< _CharT, _Traits > * __tiestr ) [inline,  
inherited]`

Ties this stream to an output stream.

**Parameters**

*tiestr* The output stream.

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see [tie\(\)](#) for more.

Definition at line 299 of file `basic_ios.h`.

**5.378.5.75** `template<typename _CharT, typename _Traits > basic_istream<  
_CharT, _Traits > & std::basic_istream< _CharT, _Traits >::unget  
( void ) [inherited]`

Unextracting the previous character.

### Returns

`*this`

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

### Note

Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 748 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::rdstate()`, `std::basic_ios< _CharT, _Traits >::setstate()`, and `std::basic_streambuf< _CharT, _Traits >::sungetc()`.

**5.378.5.76** `void std::ios_base::unsetf( fmtflags __mask ) [inline,  
inherited]`

Clearing format flags.

### Parameters

*mask* The flags to unset.

This function clears *mask* in the format flags.

Definition at line 612 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

**5.378.5.77** `template<typename _CharT, typename _Traits> char_type  
std::basic_ios< _CharT, _Traits >::widen ( char __c ) const  
[inline, inherited]`

Widens characters.

#### Parameters

*c* The character to widen.

#### Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 441 of file `basic_ios.h`.

Referenced by `std::endl()`, `std::basic_ios< char, _Traits >::fill()`, `std::basic_istream< char >::get()`, `std::basic_istream< char >::getline()`, `std::getline()`, and `std::operator>>()`.

**5.378.5.78** `streamsize std::ios_base::width ( ) const [inline,  
inherited]`

Flags access.

#### Returns

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 646 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::operator>>()`.



**5.378.5.79** streamsize std::ios\_base::width ( streamsize \_\_wide ) [inline, inherited]

Changing flags.

**Parameters**

*wide* The new width value.

**Returns**

The previous value of [width\(\)](#).

Definition at line 655 of file ios\_base.h.

**5.378.5.80** static int std::ios\_base::xalloc ( ) throw () [static, inherited]

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

**5.378.6 Member Data Documentation**

**5.378.6.1** template<typename \_CharT, typename \_Traits> streamsize std::basic\_istream< \_CharT, \_Traits >::M\_gcount [protected, inherited]

The number of characters extracted in the previous unformatted function; see [gcount\(\)](#).

Definition at line 81 of file istream.

## 5.378 `std::basic_ifstream< _CharT, _Traits >` Class Template Reference 1775

Referenced by `std::basic_istream< char >::gcount()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::basic_istream< char >::~~basic_istream()`.

### 5.378.6.2 `const fmtflags std::ios_base::adjustfield` [**static, inherited**]

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 312 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

### 5.378.6.3 `const openmode std::ios_base::app` [**static, inherited**]

Seek to end before each write.

Definition at line 366 of file `ios_base.h`.

### 5.378.6.4 `const openmode std::ios_base::ate` [**static, inherited**]

Open and seek to end immediately after opening.

Definition at line 369 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

### 5.378.6.5 `const iostate std::ios_base::badbit` [**static, inherited**]

Indicates a loss of integrity in an input or output sequence (such /// as an irrecoverable read error from a file).

Definition at line 336 of file `ios_base.h`.

Referenced by `std::basic_ostream< char >::_M_write()`, `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::basic_ios< char,`

`_Traits >::fail()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ios< _CharT, _Traits >::init()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, `std::basic_istream< _CharT, _Traits >::unget()`, `std::basic_ostream< _CharT, _Traits >::write()`, and `std::basic_ostream< _CharT, _Traits >::sentry::~sentry()`.

#### 5.378.6.6 `const fmtflags std::ios_base::basefield` [`static`, `inherited`]

A mask of `dec|oct|hex`. Useful for the 2-arg form of `setf`.

Definition at line 315 of file `ios_base.h`.

Referenced by `std::dec()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::hex()`, `std::oct()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

#### 5.378.6.7 `const seekdir std::ios_base::beg` [`static`, `inherited`]

Request a seek relative to the beginning of the stream.

Definition at line 398 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::seekpos()`.

#### 5.378.6.8 `const openmode std::ios_base::binary` [`static`, `inherited`]

Perform input and output in binary mode (as opposed to text mode). `/// This is probably not what you think it is; see http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html.`

Definition at line 374 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::showmanyc()`.

**5.378.6.9 const fmtflags std::ios\_base::boolalpha [static, inherited]**

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 260 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::noboolalpha()`.

**5.378.6.10 const seekdir std::ios\_base::cur [static, inherited]**

Request a seek relative to the current position within the sequence.

Definition at line 401 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_filebuf< _CharT, _Traits >::seekoff()`, `std::basic_istream< _CharT, _Traits >::tellg()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

**5.378.6.11 const fmtflags std::ios\_base::dec [static, inherited]**

Converts integer input or generates integer output in decimal base.

Definition at line 263 of file `ios_base.h`.

Referenced by `std::dec()`.

**5.378.6.12 const seekdir std::ios\_base::end [static, inherited]**

Request a seek relative to the current end of the sequence.

Definition at line 404 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`.

**5.378.6.13 const iostate std::ios\_base::eofbit [static, inherited]**

Indicates that an input operation reached the end of an input sequence.

Definition at line 339 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_date(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::eof(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::unget(), and std::ws().

#### 5.378.6.14 const iostate std::ios\_base::failbit [static, inherited]

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 344 of file ios\_base.h.

Referenced by std::basic\_fstream< \_CharT, \_Traits >::close(), std::basic\_ofstream< \_CharT, \_Traits >::close(), std::basic\_ifstream< \_CharT, \_Traits >::close(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::fail(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_fstream< \_CharT, \_Traits >::open(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry(), and std::basic\_istream< \_CharT, \_Traits >::sentry::sentry().

#### 5.378.6.15 const fmtflags std::ios\_base::fixed [static, inherited]

Generate floating-point output in fixed-point notation.

Definition at line 266 of file ios\_base.h.

Referenced by `std::fixed()`.

**5.378.6.16 const fmtflags std::ios\_base::floatfield [static, inherited]**

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 318 of file `ios_base.h`.

Referenced by `std::fixed()`, and `std::scientific()`.

**5.378.6.17 const iostate std::ios\_base::goodbit [static, inherited]**

Indicates all is well.

Definition at line 347 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ios< _CharT, _Traits >::init()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::basic_istream< _CharT, _Traits >::sync()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

**5.378.6.18 const fmtflags std::ios\_base::hex [static, inherited]**

Converts integer input or generates integer output in hexadecimal base.

Definition at line 269 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::hex()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

**5.378.6.19 `const openmode std::ios_base::in` [static, inherited]**

Open for input. Default for `ifstream` and `fstream`.

Definition at line 377 of file `ios_base.h`.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_set_buffer()`, `std::basic_ifstream< _CharT, _Traits >::open()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

**5.378.6.20 `const fmtflags std::ios_base::internal` [static, inherited]**

Adds fill characters at a designated internal point in certain `///` generated output, or identical to `right` if no such point is `///` designated.

Definition at line 274 of file `ios_base.h`.

Referenced by `std::internal()`.

**5.378.6.21 `const fmtflags std::ios_base::left` [static, inherited]**

Adds fill characters on the right (final positions) of certain `///` generated output. (I.e., the thing you print is flush left.).

Definition at line 278 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, and `std::left()`.

**5.378.6.22 `const fmtflags std::ios_base::oct` [static, inherited]**

Converts integer input or generates integer output in octal base.

Definition at line 281 of file `ios_base.h`.

Referenced by `std::oct()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

**5.378.6.23 const openmode std::ios\_base::out [static, inherited]**

Open for output. Default for `ofstream` and `fstream`.

Definition at line 380 of file `ios_base.h`.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_set_buffer()`, `std::basic_ofstream< _CharT, _Traits >::open()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

**5.378.6.24 const fmtflags std::ios\_base::right [static, inherited]**

Adds fill characters on the left (initial positions) of certain /// generated output. (I.e., the thing you print is flush right.).

Definition at line 285 of file `ios_base.h`.

Referenced by `std::right()`.

**5.378.6.25 const fmtflags std::ios\_base::scientific [static, inherited]**

Generates floating-point output in scientific notation.

Definition at line 288 of file `ios_base.h`.

Referenced by `std::scientific()`.

**5.378.6.26 const fmtflags std::ios\_base::showbase [static, inherited]**

Generates a prefix indicating the numeric base of generated integer /// output.

Definition at line 292 of file `ios_base.h`.

Referenced by `std::noshowbase()`, and `std::showbase()`.



**5.378.6.27 const fmtflags std::ios\_base::showpoint [static, inherited]**

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 296 of file ios\_base.h.

Referenced by std::noshowpoint(), and std::showpoint().

**5.378.6.28 const fmtflags std::ios\_base::showpos [static, inherited]**

Generates a + sign in non-negative generated numeric output.

Definition at line 299 of file ios\_base.h.

Referenced by std::noshowpos(), and std::showpos().

**5.378.6.29 const fmtflags std::ios\_base::skipws [static, inherited]**

Skips leading white space before certain input operations.

Definition at line 302 of file ios\_base.h.

Referenced by std::noskipws(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), and std::skipws().

**5.378.6.30 const openmode std::ios\_base::trunc [static, inherited]**

Open for input. Default for ofstream.

Definition at line 383 of file ios\_base.h.

**5.378.6.31 const fmtflags std::ios\_base::unitbuf [static, inherited]**

Flushes output after each output operation.

Definition at line 305 of file ios\_base.h.

Referenced by std::nounitbuf(), std::unitbuf(), and std::basic\_ostream< \_CharT, \_Traits >::sentry::~sentry().

5.378.6.32 `const fmtflags std::ios_base::uppercase` [`static`, `inherited`]

Replaces certain lowercase letters with their uppercase equivalents /// in generated output.

Definition at line 309 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, `std::nouppercase()`, and `std::uppercase()`.

The documentation for this class was generated from the following file:

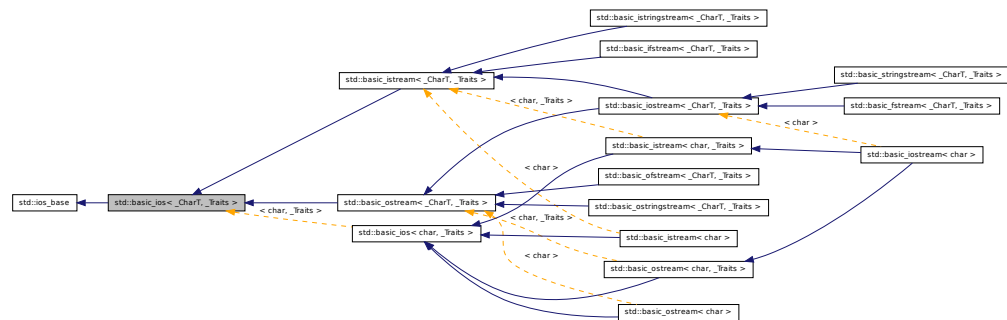
- [fstream](#)

5.379 `std::basic_ios< _CharT, _Traits >` Class Template Reference

Virtual base class for all stream classes.

Most of the member functions called dispatched on stream objects (e.g., `std::cout.foo(bar);`) are consolidated in this class.

Inheritance diagram for `std::basic_ios< _CharT, _Traits >`:



## Public Types

- enum [event](#) { `erase_event`, `imbue_event`, `copyfmt_event` }
- typedef `void(* event_callback)(event, ios_base &, int)`
- typedef `_Ios_Fmtflags fmtflags`
- typedef `int io_state`
- typedef `_Ios_Iostate iostate`

- typedef int **open\_mode**
  - typedef `_Ios_Openmode` [openmode](#)
  - typedef int **seek\_dir**
  - typedef `_Ios_Seekdir` [seekdir](#)
  - typedef `std::streamoff` **streamoff**
  - typedef `std::streampos` **streampos**
- 
- typedef `_CharT` [char\\_type](#)
  - typedef `_Traits::int_type` [int\\_type](#)
  - typedef `_Traits::pos_type` [pos\\_type](#)
  - typedef `_Traits::off_type` [off\\_type](#)
  - typedef `_Traits` [traits\\_type](#)
- 
- typedef `ctype<_CharT>` [\\_\\_ctype\\_type](#)
  - typedef `num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>>` [\\_\\_num\\_put\\_type](#)
  - typedef `num_get<_CharT, istreambuf_iterator<_CharT, _Traits>>` [\\_\\_num\\_get\\_type](#)

### Public Member Functions

- [basic\\_ios](#) ([basic\\_streambuf](#)< `_CharT`, `_Traits` > \*\_\_sb)
- virtual `~basic_ios` ()
- const [locale](#) & [\\_M\\_getloc](#) () const
- void [\\_M\\_setstate](#) ([iostate](#) \_\_state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) \_\_state=[goodbit](#))
- [basic\\_ios](#) & [copyfmt](#) (const [basic\\_ios](#) &\_\_rhs)
- bool [eof](#) () const
- [iostate](#) [exceptions](#) () const
- void [exceptions](#) ([iostate](#) \_\_except)
- bool [fail](#) () const
- [char\\_type](#) [fill](#) () const
- [char\\_type](#) [fill](#) ([char\\_type](#) \_\_ch)
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) \_\_fmtfl)
- [locale](#) [getloc](#) () const
- bool [good](#) () const
- [locale](#) [imbue](#) (const [locale](#) &\_\_loc)
- long & [iword](#) (int \_\_ix)
- char [narrow](#) ([char\\_type](#) \_\_c, char \_\_dfault) const
- [streamsize](#) [precision](#) () const
- [streamsize](#) [precision](#) ([streamsize](#) \_\_prec)

- void \*& pword (int \_\_ix)
  - basic\_streambuf< \_CharT, \_Traits > \* rdbuf (basic\_streambuf< \_CharT, \_Traits > \*\_\_sb)
  - basic\_streambuf< \_CharT, \_Traits > \* rdbuf () const
  - iostate rdstate () const
  - void register\_callback (event\_callback \_\_fn, int \_\_index)
  - fmtflags setf (fmtflags \_\_fmtfl, fmtflags \_\_mask)
  - fmtflags setf (fmtflags \_\_fmtfl)
  - void setstate (iostate \_\_state)
  - basic\_ostream< \_CharT, \_Traits > \* tie () const
  - basic\_ostream< \_CharT, \_Traits > \* tie (basic\_ostream< \_CharT, \_Traits > \*\_\_\_tiestr)
  - void unsetf (fmtflags \_\_mask)
  - char\_type widen (char \_\_c) const
  - streamsize width () const
  - streamsize width (streamsize \_\_wide)
- 
- operator void \* () const
  - bool operator! () const

### Static Public Member Functions

- static bool sync\_with\_stdio (bool \_\_sync=true)
- static int xalloc () throw ()

### Static Public Attributes

- static const fmtflags adjustfield
- static const openmode app
- static const openmode ate
- static const iostate badbit
- static const fmtflags basefield
- static const seekdir beg
- static const openmode binary
- static const fmtflags boolalpha
- static const seekdir cur
- static const fmtflags dec
- static const seekdir end
- static const iostate eofbit
- static const iostate failbit
- static const fmtflags fixed
- static const fmtflags floatfield

- static const `iostate` `goodbit`
- static const `fmtflags` `hex`
- static const `openmode` `in`
- static const `fmtflags` `internal`
- static const `fmtflags` `left`
- static const `fmtflags` `oct`
- static const `openmode` `out`
- static const `fmtflags` `right`
- static const `fmtflags` `scientific`
- static const `fmtflags` `showbase`
- static const `fmtflags` `showpoint`
- static const `fmtflags` `showpos`
- static const `fmtflags` `skipws`
- static const `openmode` `trunc`
- static const `fmtflags` `unitbuf`
- static const `fmtflags` `uppercase`

### Protected Types

- enum { `_S_local_word_size` }

### Protected Member Functions

- `basic_ios` ()
- void `_M_cache_locale` (const `locale` &\_\_loc)
- void `_M_call_callbacks` (`event` \_\_ev) throw ()
- void `_M_dispose_callbacks` (void) throw ()
- `_Words` & `_M_grow_words` (int \_\_index, bool \_\_iword)
- void `_M_init` () throw ()
- void `init` (`basic_streambuf`< `_CharT`, `_Traits` > \*\_\_sb)

### Protected Attributes

- `_Callback_list` \* `_M_callbacks`
- const `__ctype_type` \* `_M_ctype`
- `iostate` `_M_exception`
- `char_type` `_M_fill`
- bool `_M_fill_init`
- `fmtflags` `_M_flags`
- `locale` `_M_ios_locale`
- `_Words` `_M_local_word` [`_S_local_word_size`]
- const `__num_get_type` \* `_M_num_get`

- `const __num_put_type * _M_num_put`
- `streamsize _M_precision`
- `basic_streambuf<_CharT, _Traits> * _M_streambuf`
- `iosstate _M_streambuf_state`
- `basic_ostream<_CharT, _Traits> * _M_tie`
- `streamsize _M_width`
- `_Words * _M_word`
- `int _M_word_size`
- `_Words _M_word_zero`

### 5.379.1 Detailed Description

`template<typename _CharT, typename _Traits> class std::basic_ios<_CharT, _Traits>`

Virtual base class for all stream classes.

Most of the member functions called dispatched on stream objects (e.g., `std::cout.foo(bar);`) are consolidated in this class.

Definition at line 64 of file `basic_ios.h`.

### 5.379.2 Member Typedef Documentation

**5.379.2.1** `template<typename _CharT, typename _Traits> typedef ctype<_CharT> std::basic_ios<_CharT, _Traits>::__ctype_type`

These are non-standard types.

Reimplemented in `std::basic_istream<_CharT, _Traits>`, `std::basic_ostream<_CharT, _Traits>`, `std::basic_istream<char, _Traits>`, `std::basic_istream<char>`, `std::basic_ostream<char, _Traits>`, and `std::basic_ostream<char>`.

Definition at line 84 of file `basic_ios.h`.

**5.379.2.2** `template<typename _CharT, typename _Traits> typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits>> std::basic_ios<_CharT, _Traits>::__num_get_type`

These are non-standard types.

Reimplemented in `std::basic_istream<_CharT, _Traits>`, `std::basic_istream<char, _Traits>`, and `std::basic_istream<char>`.

Definition at line 88 of file `basic_ios.h`.

**5.379.2.3** `template<typename _CharT, typename _Traits> typedef  
num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> >  
std::basic_ios< _CharT, _Traits >::__num_put_type`

These are non-standard types.

Reimplemented in [std::basic\\_ostream< \\_CharT, \\_Traits >](#), [std::basic\\_ostream< char, \\_Traits >](#), and [std::basic\\_ostream< char >](#).

Definition at line 86 of file `basic_ios.h`.

**5.379.2.4** `template<typename _CharT, typename _Traits> typedef _CharT  
std::basic_ios< _CharT, _Traits >::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in [std::basic\\_ifstream< \\_CharT, \\_Traits >](#), [std::basic\\_ofstream< \\_CharT, \\_Traits >](#), [std::basic\\_fstream< \\_CharT, \\_Traits >](#), [std::basic\\_istream< \\_CharT, \\_Traits >](#), [std::basic\\_iostream< \\_CharT, \\_Traits >](#), [std::basic\\_ostream< \\_CharT, \\_Traits >](#), [std::basic\\_istreamstream< \\_CharT, \\_Traits, \\_Alloc >](#), [std::basic\\_ostreamstream< \\_CharT, \\_Traits, \\_Alloc >](#), [std::basic\\_stringstream< \\_CharT, \\_Traits, \\_Alloc >](#), [std::basic\\_istream< char, \\_Traits >](#), [std::basic\\_istream< char >](#), [std::basic\\_iostream< char >](#), [std::basic\\_ostream< char, \\_Traits >](#), and [std::basic\\_ostream< char >](#).

Definition at line 73 of file `basic_ios.h`.

**5.379.2.5** `typedef void(* std::ios_base::event_callback)(event, ios_base &, int)  
[inherited]`

The type of an event callback function.

### Parameters

*event* One of the members of the event enum.

*ios\_base* Reference to the [ios\\_base](#) object.

*int* The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several [ios\\_base](#) and [basic\\_ios](#) functions, specifically [imbue\(\)](#), [copyfmt\(\)](#), and [~ios\(\)](#).

Definition at line 438 of file `ios_base.h`.

### 5.379.2.6 `typedef _Ios_Fmtflags std::ios_base::fmtflags` [inherited]

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 257 of file `ios_base.h`.



### 5.379.2.7 `template<typename _CharT, typename _Traits> typedef _Traits::int_type std::basic_ios<_CharT, _Traits>::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in `std::basic_ifstream<_CharT, _Traits>`, `std::basic_ofstream<_CharT, _Traits>`, `std::basic_fstream<_CharT, _Traits>`, `std::basic_istream<_CharT, _Traits>`, `std::basic_iostream<_CharT, _Traits>`, `std::basic_ostream<_CharT, _Traits>`, `std::basic_istreamstream<_CharT, _Traits, _Alloc>`, `std::basic_ostreamstream<_CharT, _Traits, _Alloc>`, `std::basic_stringstream<_CharT, _Traits, _Alloc>`, `std::basic_istream<char, _Traits>`, `std::basic_istream<char>`, `std::basic_iostream<char>`, `std::basic_ostream<char, _Traits>`, and `std::basic_ostream<char>`.

Definition at line 74 of file `basic_ios.h`.

### 5.379.2.8 `typedef _Ios_Iostate std::ios_base::iostate [inherited]`

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 332 of file `ios_base.h`.

### 5.379.2.9 `template<typename _CharT, typename _Traits> typedef _Traits::off_type std::basic_ios<_CharT, _Traits>::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in `std::basic_ifstream<_CharT, _Traits>`, `std::basic_ofstream<_CharT, _Traits>`, `std::basic_fstream<_CharT, _Traits>`, `std::basic_istream<_CharT, _Traits>`, `std::basic_iostream<_CharT, _Traits>`, `std::basic_ostream<_CharT, _Traits>`, `std::basic_istreamstream<_CharT, _Traits, _Alloc>`, `std::basic_ostreamstream<_CharT, _Traits, _Alloc>`, `std::basic_stringstream<_CharT, _Traits, _Alloc>`, `std::basic_istream<char, _Traits>`, `std::basic_istream<char>`, `std::basic_iostream<char>`, `std::basic_ostream<char, _Traits>`, and `std::basic_ostream<char>`.

`CharT, _Traits >`, `std::basic_istream<_CharT, _Traits, _Alloc >`, `std::basic_ostringstream<_CharT, _Traits, _Alloc >`, `std::basic_stringstream<_CharT, _Traits, _Alloc >`, `std::basic_istream<char, _Traits >`, `std::basic_istream<char >`, `std::basic_iostream<char >`, `std::basic_ostream<char, _Traits >`, and `std::basic_ostream<char >`.

Definition at line 76 of file `basic_ios.h`.

#### 5.379.2.10 `typedef _Ios_Openmode std::ios_base::openmode` [inherited]

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 363 of file `ios_base.h`.

#### 5.379.2.11 `template<typename _CharT, typename _Traits> typedef _Traits::pos_type std::basic_ios<_CharT, _Traits>::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in `std::basic_ifstream<_CharT, _Traits >`, `std::basic_ofstream<_CharT, _Traits >`, `std::basic_fstream<_CharT, _Traits >`, `std::basic_istream<_CharT, _Traits >`, `std::basic_iostream<_CharT, _Traits >`, `std::basic_ostream<_CharT, _Traits >`, `std::basic_istream<char, _Traits >`, `std::basic_istream<char >`, `std::basic_iostream<char >`, `std::basic_ostream<char, _Traits >`, and `std::basic_ostream<char >`.

Definition at line 75 of file `basic_ios.h`.

### 5.379.2.12 `typedef _Ios_Seekdir std::ios_base::seekdir` [inherited]

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 395 of file `ios_base.h`.

### 5.379.2.13 `template<typename _CharT, typename _Traits> typedef _Traits std::basic_ios<_CharT, _Traits>::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in `std::basic_ifstream<_CharT, _Traits>`, `std::basic_ofstream<_CharT, _Traits>`, `std::basic_fstream<_CharT, _Traits>`, `std::basic_istream<_CharT, _Traits>`, `std::basic_iostream<_CharT, _Traits>`, `std::basic_ostream<_CharT, _Traits>`, `std::basic_istreamstream<_CharT, _Traits, _Alloc>`, `std::basic_ostreamstream<_CharT, _Traits, _Alloc>`, `std::basic_stringstream<_CharT, _Traits, _Alloc>`, `std::basic_istream<char, _Traits>`, `std::basic_istream<char>`, `std::basic_iostream<char>`, `std::basic_ostream<char, _Traits>`, and `std::basic_ostream<char>`.

Definition at line 77 of file `basic_ios.h`.

## 5.379.3 Member Enumeration Documentation

### 5.379.3.1 `enum std::ios_base::event` [inherited]

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 421 of file `ios_base.h`.

#### 5.379.4 Constructor & Destructor Documentation

**5.379.4.1** `template<typename _CharT, typename _Traits> std::basic_ios<_CharT, _Traits >::basic_ios ( basic_streambuf< _CharT, _Traits >* __sb ) [inline, explicit]`

Constructor performs initialization.

The parameter is passed by derived streams.

Definition at line 262 of file basic\_ios.h.

**5.379.4.2** `template<typename _CharT, typename _Traits> virtual std::basic_ios< _CharT, _Traits >::~~basic_ios ( ) [inline, virtual]`

Empty.

The destructor does nothing. More specifically, it does not destroy the streambuf held by [rdbuf\(\)](#).

Definition at line 274 of file basic\_ios.h.

**5.379.4.3** `template<typename _CharT, typename _Traits> std::basic_ios<_CharT, _Traits >::basic_ios ( ) [inline, protected]`

Empty.

The default constructor does nothing and is not normally accessible to users.

Definition at line 452 of file basic\_ios.h.

#### 5.379.5 Member Function Documentation

**5.379.5.1** `const locale& std::ios_base::_M_getloc ( ) const [inline, inherited]`

Locale access.

##### Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 708 of file `ios_base.h`.

Referenced by `std::money_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_time()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::time_put< _CharT, _OutIter >::do_put()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::time_put< _CharT, _OutIter >::put()`.

#### 5.379.5.2 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::bad ( ) const [inline]`

Fast error checking.

##### Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 203 of file `basic_ios.h`.

#### 5.379.5.3 `template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::clear ( iostate __state = goodbit )`

[Re]sets the error state.

##### Parameters

*state* The new state flag(s) to set.

See [std::ios\\_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

Definition at line 42 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::exceptions()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::rdstate()`.

Referenced by `std::basic_ios< char, _Traits >::exceptions()`, `std::basic_fstream< _CharT, _Traits >::open()`, `std::basic_ofstream< _CharT, _Traits >::open()`, `std::basic_ifstream< _CharT, _Traits >::open()`, `std::basic_istream< _CharT, _Traits`

>::putback(), std::basic\_ios< \_CharT, \_Traits >::rdbuf(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ios< char, \_Traits >::setstate(), and std::basic\_istream< \_CharT, \_Traits >::unget().

**5.379.5.4** `template<typename _CharT, typename _Traits> basic_ios< _CharT, _Traits> & std::basic_ios< _CharT, _Traits>::copyfmt ( const basic_ios< _CharT, _Traits> & __rhs )`

Copies fields of `__rhs` into this.

#### Parameters

`__rhs` The source values for the copies.

#### Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy `exceptions()`.

Definition at line 64 of file `basic_ios.tcc`.

References `std::basic_ios< _CharT, _Traits>::exceptions()`, `std::basic_ios< _CharT, _Traits>::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios< _CharT, _Traits>::tie()`, and `std::ios_base::width()`.

**5.379.5.5** `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits>::eof ( ) const [inline]`

Fast error checking.

#### Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 182 of file `basic_ios.h`.

Referenced by `std::basic_istream< _CharT, _Traits>::get()`, `std::basic_istream< _CharT, _Traits>::getline()`, `std::basic_istream< _CharT, _Traits>::ignore()`,

`std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

**5.379.5.6** `template<typename _CharT, typename _Traits> iostate  
std::basic_ios< _CharT, _Traits >::exceptions ( ) const [inline]`

Throwing exceptions on errors.

### Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of [exceptions\(iostate\)](#) for the meaning of the return value.

Definition at line 214 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::clear()`, and `std::basic_ios< _CharT, _Traits >::copyfmt()`.

**5.379.5.7** `template<typename _CharT, typename _Traits> void std::basic_ios<  
_CharT, _Traits >::exceptions ( iostate __except ) [inline]`

Throwing exceptions on errors.

### Parameters

*except* The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type [std::ios\\_base::failure](#) is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
 std::set_terminate (__gnu_cxx::__verbose_terminate_handler);
```

```

std::ifstream f ("/etc/motd");

std::cerr << "Setting badbit\n";
f.setstate (std::ios_base::badbit);

std::cerr << "Setting exception mask\n";
f.exceptions (std::ios_base::badbit);
}

```

Definition at line 249 of file basic\_ios.h.

#### 5.379.5.8 template<typename \_CharT, typename \_Traits> bool std::basic\_ios<\_CharT, \_Traits>::fail( ) const [inline]

Fast error checking.

##### Returns

True if either the badbit or the failbit is set.

Checking the badbit in [fail\(\)](#) is historical practice. Note that other iostate flags may also be set.

Definition at line 193 of file basic\_ios.h.

Referenced by std::basic\_ios<char, \_Traits>::operator void \*(), std::basic\_ios<char, \_Traits>::operator!(), std::basic\_istream<\_CharT, \_Traits>::seekg(), std::basic\_ostream<\_CharT, \_Traits>::seekp(), std::basic\_istream<\_CharT, \_Traits>::tellg(), std::basic\_ostream<\_CharT, \_Traits>::tellp(), and std::regex\_traits<\_Ch\_type>::value().

#### 5.379.5.9 template<typename \_CharT, typename \_Traits> char\_type std::basic\_ios<\_CharT, \_Traits>::fill( ) const [inline]

Retrieves the *empty* character.

##### Returns

The current fill character.

It defaults to a space ( ' ' ) in the current locale.

Definition at line 362 of file basic\_ios.h.

Referenced by std::basic\_ios<\_CharT, \_Traits>::copyfmt(), and std::basic\_ios<char, \_Traits>::fill().



**5.379.5.10** `template<typename _CharT, typename _Traits> char_type  
std::basic_ios< _CharT, _Traits >::fill ( char_type __ch )  
[inline]`

Sets a new *empty* character.

#### Parameters

*ch* The new character.

#### Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 382 of file `basic_ios.h`.

**5.379.5.11** `fmtflags std::ios_base::flags ( ) const [inline, inherited]`

Access to format flags.

#### Returns

The format control flags for both input and output.

Definition at line 553 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, `std::operator>>()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

**5.379.5.12** `fmtflags std::ios_base::flags ( fmtflags __fmtfl ) [inline,  
inherited]`

Setting new format flags all at once.

#### Parameters

*fmtfl* The new flags to set.

**Returns**

The previous format control flags.

This function overwrites all the format flags with *fmtfl*.

Definition at line 564 of file ios\_base.h.

**5.379.5.13 locale std::ios\_base::getloc ( ) const [inline, inherited]**

Locale access.

**Returns**

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 697 of file ios\_base.h.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::money_put< _CharT, _Outputter >::do_put()`, `std::basic_ios< _CharT, _Traits >::imbue()`, `std::operator>>()`, and `std::ws()`.

**5.379.5.14 template<typename \_CharT, typename \_Traits> bool  
std::basic\_ios< \_CharT, \_Traits >::good ( ) const [inline]**

Fast error checking.

**Returns**

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 172 of file basic\_ios.h.

Referenced by `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

**5.379.5.15 template<typename \_CharT, typename \_Traits> locale  
std::basic\_ios< \_CharT, \_Traits >::imbue ( const locale & \_\_loc )**

Moves to a new locale.

### Parameters

*loc* The new locale.

### Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Reimplemented from `std::ios_base`.

Definition at line 115 of file `basic_ios.tcc`.

References `std::ios_base::getloc()`, and `std::basic_ios< _CharT, _Traits >::rdbuf()`.

Referenced by `std::operator<<()`.

**5.379.5.16** `template<typename _CharT, typename _Traits> void  
std::basic_ios< _CharT, _Traits >::init ( basic_streambuf<  
_CharT, _Traits > * __sb ) [protected]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 127 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

Referenced by `std::basic_fstream< _CharT, _Traits >::basic_fstream()`, `std::basic_ifstream< _CharT, _Traits >::basic_ifstream()`, `std::basic_ios< char, _Traits >::basic_ios()`, `std::basic_istream< char >::basic_istream()`, `std::basic_istreamstream< _CharT, _Traits, _Alloc >::basic_istreamstream()`, `std::basic_ofstream< _CharT, _Traits >::basic_ofstream()`, `std::basic_ostream< char >::basic_ostream()`, `std::basic_ostreamstream< _CharT, _Traits, _Alloc >::basic_ostreamstream()`, and `std::basic_stringstream< _CharT, _Traits, _Alloc >::basic_stringstream()`.

**5.379.5.17** `long& std::ios_base::iword ( int __ix ) [inline, inherited]`

Access to integer array.

**Parameters**

`__ix` Index into the array.

**Returns**

A reference to an integer associated with the index.

The `ix` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 743 of file `ios_base.h`.

**5.379.5.18** `template<typename _CharT, typename _Traits> char  
std::basic_ios<_CharT, _Traits>::narrow ( char_type __c, char  
__dfault ) const [inline]`

Squeezes characters.

**Parameters**

`c` The character to narrow.

`dfault` The character to narrow.

**Returns**

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, dfault)
```

Additional notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 422 of file `basic_ios.h`.

**5.379.5.19** `template<typename _CharT, typename _Traits> std::basic_ios<  
_CharT, _Traits>::operator void * ( ) const [inline]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 113 of file `basic_ios.h`.

```
5.379.5.20 template<typename _CharT, typename _Traits> bool
std::basic_ios<_CharT, _Traits>::operator! () const
[inline]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 117 of file `basic_ios.h`.

```
5.379.5.21 streamsize std::ios_base::precision () const [inline,
inherited]
```

Flags access.

### Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 623 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, and `std::operator<<()`.

```
5.379.5.22 streamsize std::ios_base::precision (streamsize __prec)
[inline, inherited]
```

Changing flags.

### Parameters

*prec* The new precision value.

**Returns**

The previous value of `precision()`.

Definition at line 632 of file `ios_base.h`.

**5.379.5.23** `void*& std::ios_base::pword ( int __ix ) [inline, inherited]`

Access to void pointer array.

**Parameters**

`__ix` Index into the array.

**Returns**

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 764 of file `ios_base.h`.

**5.379.5.24** `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rdbuf ( basic_streambuf<_CharT, _Traits> * __sb )`

Changing the underlying buffer.

**Parameters**

`sb` The new stream buffer.

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rddbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream foo; // or some other derived type
std::streambuf* p =;

foo.ios::rddbuf(p); // ios == basic_ios<char>
```

Definition at line 54 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::clear()`.

**5.379.5.25** `template<typename _CharT, typename _Traits>`  
`basic_streambuf<_CharT, _Traits>* std::basic_ios<_CharT,`  
`_Traits>::rddbuf( ) const [inline]`

Accessing the underlying buffer.

### Returns

The current stream buffer.

This does not change the state of the stream.

Reimplemented in `std::basic_ifstream<_CharT, _Traits>`, `std::basic_ofstream<_CharT, _Traits>`, `std::basic_fstream<_CharT, _Traits>`, `std::basic_istreamstream<_CharT, _Traits, _Alloc>`, `std::basic_ostringstream<_CharT, _Traits, _Alloc>`, and `std::basic_stringstream<_CharT, _Traits, _Alloc>`.

Definition at line 313 of file `basic_ios.h`.

Referenced by `std::basic_ostream<char>::_M_write()`, `std::basic_ios<_CharT, _Traits>::clear()`, `std::basic_ostream<_CharT, _Traits>::flush()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_ios<_CharT, _Traits>::imbue()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_ostream<_CharT, _Traits>::put()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::readsome()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, `std::basic_istream<_CharT, _Traits>::sync()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_ostream<_CharT, _Traits>::tellp()`, `std::basic_istream<_CharT, _Traits>::unset()`, and `std::ws()`.

**5.379.5.26** `template<typename _CharT, typename _Traits> iostate  
std::basic_ios< _CharT, _Traits >::rdstate ( ) const [inline]`

Returns the error state of the stream buffer.

#### Returns

A bit pattern (well, isn't everything?)

See [std::ios\\_base::iostate](#) for the possible bit values. Most users will call one of the interpreting wrappers, e.g., [good\(\)](#).

Definition at line 129 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::basic_ios< char, _Traits >::eof()`, `std::basic_ios< char, _Traits >::fail()`, `std::basic_ios< char, _Traits >::good()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ios< char, _Traits >::setstate()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

**5.379.5.27** `void std::ios_base::register_callback ( event_callback __fn, int  
__index ) [inherited]`

Add the callback `__fn` with parameter `__index`.

#### Parameters

*`__fn`* The function to add.

*`__index`* The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**5.379.5.28** `fmtflags std::ios_base::setf ( fmtflags __fmtfl ) [inline,  
inherited]`

Setting new format flags.

#### Parameters

*`__fmtfl`* Additional flags to set.



**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 580 of file ios\_base.h.

Referenced by std::dec(), std::fixed(), std::hex(), std::left(), std::oct(), std::right(), std::scientific(), std::showbase(), std::showpoint(), std::showpos(), std::skipws(), std::unitbuf(), and std::uppercase().

**5.379.5.29** `fmtflags std::ios_base::setf ( fmtflags __fmtfl, fmtflags __mask )`  
`[inline, inherited]`

Setting new format flags.

**Parameters**

*fmtfl* Additional flags to set.

*mask* The flags mask for *fmtfl*.

**Returns**

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 597 of file ios\_base.h.

**5.379.5.30** `template<typename _CharT, typename _Traits> void`  
`std::basic_ios<_CharT, _Traits>::setstate ( iostate __state )`  
`[inline]`

Sets additional flags in the error state.

**Parameters**

*state* The additional state flag(s) to set.

See `std::ios_base::iostate` for the possible bit values.

Definition at line 149 of file basic\_ios.h.

Referenced by std::basic\_ostream< char >::\_M\_write(), std::basic\_fstream< \_CharT, \_Traits >::close(), std::basic\_ofstream< \_CharT, \_Traits >::close(), std::basic\_ifstream< \_CharT, \_Traits >::close(), std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_fstream< \_CharT, \_Traits >::open(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::sync(), std::basic\_istream< \_CharT, \_Traits >::unget(), and std::ws().

#### 5.379.5.31 static bool std::ios\_base::sync\_with\_stdio ( bool \_\_sync = true ) [static, inherited]

Interaction with the standard C I/O objects.

##### Parameters

*sync* Whether to synchronize or not.

##### Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

#### 5.379.5.32 template<typename \_CharT, typename \_Traits> basic\_ostream<\_CharT, \_Traits>\* std::basic\_ios< \_CharT, \_Traits >::tie ( ) const [inline]

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 287 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

**5.379.5.33** `template<typename _CharT, typename _Traits>  
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie ( basic_ostream<_CharT, _Traits> * __tiestr ) [inline]`

Ties this stream to an output stream.

**Parameters**

*tiestr* The output stream.

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 299 of file `basic_ios.h`.

**5.379.5.34** `void std::ios_base::unsetf ( fmtflags __mask ) [inline, inherited]`

Clearing format flags.

**Parameters**

*mask* The flags to unset.

This function clears *mask* in the format flags.

Definition at line 612 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

**5.379.5.35** `template<typename _CharT, typename _Traits> char_type  
std::basic_ios< _CharT, _Traits >::widen ( char __c ) const  
[inline]`

Widens characters.

#### Parameters

*c* The character to widen.

#### Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.ht>

Definition at line 441 of file `basic_ios.h`.

Referenced by `std::endl()`, `std::basic_ios< char, _Traits >::fill()`, `std::basic_istream< char >::get()`, `std::basic_istream< char >::getline()`, `std::getline()`, and `std::operator>>()`.

**5.379.5.36** `streamsize std::ios_base::width ( ) const [inline,  
inherited]`

Flags access.

#### Returns

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 646 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::operator>>()`.

**5.379.5.37** `streamsize std::ios_base::width ( streamsize __wide ) [inline, inherited]`

Changing flags.

**Parameters**

*wide* The new width value.

**Returns**

The previous value of `width()`.

Definition at line 655 of file `ios_base.h`.

**5.379.5.38** `static int std::ios_base::xalloc ( ) throw () [static, inherited]`

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

**5.379.6 Member Data Documentation****5.379.6.1** `const fmtflags std::ios_base::adjustfield [static, inherited]`

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Definition at line 312 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

**5.379.6.2 `const openmode std::ios_base::app` [`static`, `inherited`]**

Seek to end before each write.

Definition at line 366 of file `ios_base.h`.

**5.379.6.3 `const openmode std::ios_base::ate` [`static`, `inherited`]**

Open and seek to end immediately after opening.

Definition at line 369 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

**5.379.6.4 `const iostate std::ios_base::badbit` [`static`, `inherited`]**

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 336 of file `ios_base.h`.

Referenced by `std::basic_ostream< char >::_M_write()`, `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::basic_ios< char, _Traits >::fail()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ios< _CharT, _Traits >::init()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, `std::basic_istream< _CharT, _Traits >::unget()`, `std::basic_ostream< _CharT, _Traits >::write()`, and `std::basic_ostream< _CharT, _Traits >::sentry::~sentry()`.

**5.379.6.5 `const fmtflags std::ios_base::basefield` [`static`, `inherited`]**

A mask of `dec|oct|hex`. Useful for the 2-arg form of `setf`.

Definition at line 315 of file `ios_base.h`.

Referenced by `std::dec()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::hex()`, `std::oct()`, and `std::basic_ostream<_CharT, _Traits>::operator<<()`.

#### 5.379.6.6 `const seekdir std::ios_base::beg` [`static`, `inherited`]

Request a seek relative to the beginning of the stream.

Definition at line 398 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::seekpos()`.

#### 5.379.6.7 `const openmode std::ios_base::binary` [`static`, `inherited`]

Perform input and output in binary mode (as opposed to text mode). /// This is probably not what you think it is; see /// <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 374 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::showmanyc()`.

#### 5.379.6.8 `const fmtflags std::ios_base::boolalpha` [`static`, `inherited`]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 260 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::noboolalpha()`.

#### 5.379.6.9 `const seekdir std::ios_base::cur` [`static`, `inherited`]

Request a seek relative to the current position within the sequence.

Definition at line 401 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::pbackfail()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_filebuf<_`

CharT, \_Traits >::seekoff(), std::basic\_istream< \_CharT, \_Traits >::tellg(), and std::basic\_ostream< \_CharT, \_Traits >::tellp().

#### 5.379.6.10 const fmtflags std::ios\_base::dec [static, inherited]

Converts integer input or generates integer output in decimal base.

Definition at line 263 of file ios\_base.h.

Referenced by std::dec().

#### 5.379.6.11 const seekdir std::ios\_base::end [static, inherited]

Request a seek relative to the current end of the sequence.

Definition at line 404 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::open(), and std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff().

#### 5.379.6.12 const iostate std::ios\_base::eofbit [static, inherited]

Indicates that an input operation reached the end of an input sequence.

Definition at line 339 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_date(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::eof(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::unset(), and std::ws().

#### 5.379.6.13 const iostate std::ios\_base::failbit [static, inherited]



Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 344 of file ios\_base.h.

Referenced by std::basic\_fstream< \_CharT, \_Traits >::close(), std::basic\_ofstream< \_CharT, \_Traits >::close(), std::basic\_ifstream< \_CharT, \_Traits >::close(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::fail(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_fstream< \_CharT, \_Traits >::open(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry(), and std::basic\_istream< \_CharT, \_Traits >::sentry::sentry().

#### 5.379.6.14 const fmtflags std::ios\_base::fixed [static, inherited]

Generate floating-point output in fixed-point notation.

Definition at line 266 of file ios\_base.h.

Referenced by std::fixed().

#### 5.379.6.15 const fmtflags std::ios\_base::floatfield [static, inherited]

A mask of scientific|fixed. Useful for the 2-arg form of setf.

Definition at line 318 of file ios\_base.h.

Referenced by std::fixed(), and std::scientific().

#### 5.379.6.16 const iostate std::ios\_base::goodbit [static, inherited]

Indicates all is well.

Definition at line 347 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_

get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_ios< \_CharT, \_Traits >::init(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::sync(), and std::basic\_istream< \_CharT, \_Traits >::unget().

#### 5.379.6.17 const fmtflags std::ios\_base::hex [static, inherited]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 269 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), std::hex(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

#### 5.379.6.18 const openmode std::ios\_base::in [static, inherited]

Open for input. Default for ifstream and fstream.

Definition at line 377 of file ios\_base.h.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::M\_set\_buffer(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_filebuf< \_CharT, \_Traits >::pbackfail(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::showmanyc(), std::basic\_filebuf< \_CharT, \_Traits >::showmanyc(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::underflow(), std::basic\_filebuf< \_CharT, \_Traits >::underflow(), and std::basic\_filebuf< \_CharT, \_Traits >::xsgetn().

#### 5.379.6.19 const fmtflags std::ios\_base::internal [static, inherited]

Adds fill characters at a designated internal point in certain `///` generated output, or identical to `right` if no such point is `///` designated.

Definition at line 274 of file `ios_base.h`.

Referenced by `std::internal()`.

#### 5.379.6.20 `const fmtflags std::ios_base::left` [`static`, `inherited`]

Adds fill characters on the right (final positions) of certain `///` generated output. (I.e., the thing you print is flush left.).

Definition at line 278 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, and `std::left()`.

#### 5.379.6.21 `const fmtflags std::ios_base::oct` [`static`, `inherited`]

Converts integer input or generates integer output in octal base.

Definition at line 281 of file `ios_base.h`.

Referenced by `std::oct()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

#### 5.379.6.22 `const openmode std::ios_base::out` [`static`, `inherited`]

Open for output. Default for `ofstream` and `fstream`.

Definition at line 380 of file `ios_base.h`.

Referenced by `std::basic_filebuf< char_type, traits_type >::M_set_buffer()`, `std::basic_ofstream< _CharT, _Traits >::open()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

#### 5.379.6.23 `const fmtflags std::ios_base::right` [`static`, `inherited`]

Adds fill characters on the left (initial positions) of certain /// generated output. (I.e., the thing you print is flush right.).

Definition at line 285 of file ios\_base.h.

Referenced by std::right().

#### **5.379.6.24 const fmtflags std::ios\_base::scientific [static, inherited]**

Generates floating-point output in scientific notation.

Definition at line 288 of file ios\_base.h.

Referenced by std::scientific().

#### **5.379.6.25 const fmtflags std::ios\_base::showbase [static, inherited]**

Generates a prefix indicating the numeric base of generated integer /// output.

Definition at line 292 of file ios\_base.h.

Referenced by std::noshowbase(), and std::showbase().

#### **5.379.6.26 const fmtflags std::ios\_base::showpoint [static, inherited]**

Generates a decimal-point character unconditionally in generated /// floating-point output.

Definition at line 296 of file ios\_base.h.

Referenced by std::noshowpoint(), and std::showpoint().

#### **5.379.6.27 const fmtflags std::ios\_base::showpos [static, inherited]**

Generates a + sign in non-negative generated numeric output.

Definition at line 299 of file ios\_base.h.

Referenced by std::noshowpos(), and std::showpos().

**5.379.6.28 const fmtflags std::ios\_base::skipws [static, inherited]**

Skips leading white space before certain input operations.

Definition at line 302 of file ios\_base.h.

Referenced by std::noskipws(), std::basic\_istream<\_CharT, \_Traits>::sentry::sentry(), and std::skipws().

**5.379.6.29 const openmode std::ios\_base::trunc [static, inherited]**

Open for input. Default for ostream.

Definition at line 383 of file ios\_base.h.

**5.379.6.30 const fmtflags std::ios\_base::unitbuf [static, inherited]**

Flushes output after each output operation.

Definition at line 305 of file ios\_base.h.

Referenced by std::nunitbuf(), std::unitbuf(), and std::basic\_ostream<\_CharT, \_Traits>::sentry::~sentry().

**5.379.6.31 const fmtflags std::ios\_base::uppercase [static, inherited]**

Replaces certain lowercase letters with their uppercase equivalents /// in generated output.

Definition at line 309 of file ios\_base.h.

Referenced by std::num\_put<\_CharT, \_OutIter>::do\_put(), std::nouppercase(), and std::uppercase().

The documentation for this class was generated from the following files:

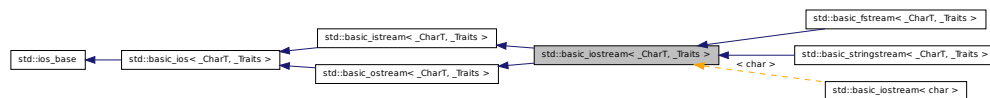
- [basic\\_ios.h](#)
- [basic\\_ios.tcc](#)

## 5.380 `std::basic_iostream< _CharT, _Traits >` Class Template Reference

Merging istream and ostream capabilities.

This class multiply inherits from the input and output stream classes simply to provide a single interface.

Inheritance diagram for `std::basic_iostream< _CharT, _Traits >`:



### Public Types

- typedef `ctype< _CharT > __ctype_type`
- typedef `ctype< _CharT > __ctype_type`
- typedef `basic_ios< _CharT, _Traits > __ios_type`
- typedef `basic_ios< _CharT, _Traits > __ios_type`
- typedef `basic_istream< _CharT, _Traits > __istream_type`
- typedef `basic_istream< _CharT, _Traits > __istream_type`
- typedef `num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > __num_get_type`
- typedef `num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __num_put_type`
- typedef `basic_ostream< _CharT, _Traits > __ostream_type`
- typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`
- typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`
- typedef `_CharT char_type`
- typedef `_CharT char_type`
- enum `event { erase_event, imbue_event, copyfmt_event }`
- typedef `void(* event_callback)(event, ios_base &, int)`
- typedef `_Ios_Fmtflags fmtflags`
- typedef `_Traits::int_type int_type`
- typedef `_Traits::int_type int_type`
- typedef `int io_state`
- typedef `_Ios_Iostate iostate`
- typedef `_Traits::off_type off_type`
- typedef `_Traits::off_type off_type`
- typedef `int open_mode`

- typedef `_Ios_Openmode` [openmode](#)
- typedef `_Traits::pos_type` [pos\\_type](#)
- typedef `_Traits::pos_type` [pos\\_type](#)
- typedef `int` [seek\\_dir](#)
- typedef `_Ios_Seekdir` [seekdir](#)
- typedef `std::streamoff` [streamoff](#)
- typedef `std::streampos` [streampos](#)
- typedef `_Traits` [traits\\_type](#)
- typedef `_Traits` [traits\\_type](#)
  
- typedef `num_put<_CharT, ostreambuf_iterator<_CharT, _Traits > > __-`  
`num_put_type`

### Public Member Functions

- [basic\\_istream](#) ([basic\\_streambuf](#)<\_CharT, \_Traits > \*\_\_sb)
- virtual [~basic\\_istream](#) ()
- const [locale](#) & [\\_M\\_getloc](#) () const
- void [\\_M\\_setstate](#) ([iostate](#) \_\_state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) \_\_state=[goodbit](#))
- [basic\\_ios](#) & [copyfmt](#) (const [basic\\_ios](#) &\_\_rhs)
- bool [eof](#) () const
- [iostate](#) [exceptions](#) () const
- void [exceptions](#) ([iostate](#) \_\_except)
- bool [fail](#) () const
- [char\\_type](#) [fill](#) () const
- [char\\_type](#) [fill](#) ([char\\_type](#) \_\_ch)
- [fmtflags](#) [flags](#) ([fmtflags](#) \_\_fmtfl)
- [fmtflags](#) [flags](#) () const
- [\\_\\_ostream\\_type](#) & [flush](#) ()
- [streamsize](#) [gcount](#) () const
- template<>  
[basic\\_istream](#)< [char](#) > & [getline](#) ([char\\_type](#) \*\_\_s, [streamsize](#) \_\_n, [char\\_type](#)  
\_\_delim)
- template<>  
[basic\\_istream](#)< [wchar\\_t](#) > & [getline](#) ([char\\_type](#) \*\_\_s, [streamsize](#) \_\_n, [char\\_type](#)  
\_\_delim)
- [locale](#) [getloc](#) () const
- bool [good](#) () const
- template<>  
[basic\\_istream](#)< [wchar\\_t](#) > & [ignore](#) ([streamsize](#) \_\_n)

- `template<>`  
`basic_istream< wchar_t > & ignore (streamsize __n, int_type __delim)`
  - `template<>`  
`basic_istream< char > & ignore (streamsize __n)`
  - `template<>`  
`basic_istream< char > & ignore (streamsize __n, int_type __delim)`
  - `locale imbue (const locale & __loc)`
  - `long & iword (int __ix)`
  - `char narrow (char_type __c, char __dfault) const`
  - `streamsize precision () const`
  - `streamsize precision (streamsize __prec)`
  - `void *& pword (int __ix)`
  - `basic_streambuf< _CharT, _Traits > * rdbuf () const`
  - `basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > * __sb)`
  - `iosstate rdstate () const`
  - `void register_callback (event_callback __fn, int __index)`
  - `__ostream_type & seekp (pos_type)`
  - `__ostream_type & seekp (off_type, ios_base::seekdir)`
  - `fmtflags setf (fmtflags __fmtfl)`
  - `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
  - `void setstate (iosstate __state)`
  - `pos_type tellp ()`
  - `basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > * __tistr)`
  - `basic_ostream< _CharT, _Traits > * tie () const`
  - `void unsetf (fmtflags __mask)`
  - `char_type widen (char __c) const`
  - `streamsize width (streamsize __wide)`
  - `streamsize width () const`
- 
- `__istream_type & operator>> (__istream_type &(__pf)(__istream_type &))`
  - `__istream_type & operator>> (__ios_type &(__pf)(__ios_type &))`
  - `__istream_type & operator>> (ios_base &(__pf)(ios_base &))`

### Arithmetic Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.



If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__istream_type & operator>> (bool &__n)`
- `__istream_type & operator>> (short &__n)`
- `__istream_type & operator>> (unsigned short &__n)`
- `__istream_type & operator>> (int &__n)`
- `__istream_type & operator>> (unsigned int &__n)`
- `__istream_type & operator>> (long &__n)`
- `__istream_type & operator>> (unsigned long &__n)`
- `__istream_type & operator>> (long long &__n)`
- `__istream_type & operator>> (unsigned long long &__n)`
- `__istream_type & operator>> (float &__f)`
- `__istream_type & operator>> (double &__f)`
- `__istream_type & operator>> (long double &__f)`
- `__istream_type & operator>> (void *&__p)`
- `__istream_type & operator>> (__streambuf_type *__sb)`

### Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `int_type get ()`
- `__istream_type & get (char_type &__c)`
- `__istream_type & get (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & get (char_type *__s, streamsize __n)`
- `__istream_type & get (__streambuf_type &__sb, char_type __delim)`
- `__istream_type & get (__streambuf_type &__sb)`
- `__istream_type & getline (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & getline (char_type *__s, streamsize __n)`
- `__istream_type & ignore ()`
- `__istream_type & ignore (streamsize __n)`
- `__istream_type & ignore (streamsize __n, int_type __delim)`
- `int_type peek ()`
- `__istream_type & read (char_type *__s, streamsize __n)`

- `streamsize` `readsome` (`char_type` \*\_\_s, `streamsize` \_\_n)
- `__istream_type` & `putback` (`char_type` \_\_c)
- `__istream_type` & `unget` ()
- `int` `sync` ()
- `pos_type` `tellg` ()
- `__istream_type` & `seekg` (`pos_type`)
- `__istream_type` & `seekg` (`off_type`, `ios_base::seekdir`)
- `operator void *` () `const`
- `bool` `operator!` () `const`
- `__ostream_type` & `operator<<` (`__ostream_type` &(\*\_\_pf)(`__ostream_type` &))
- `__ostream_type` & `operator<<` (`__ios_type` &(\*\_\_pf)(`__ios_type` &))
- `__ostream_type` & `operator<<` (`ios_base` &(\*\_\_pf)(`ios_base` &))

### Arithmetic Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__ostream_type` & `operator<<` (`long` \_\_n)
- `__ostream_type` & `operator<<` (`unsigned long` \_\_n)
- `__ostream_type` & `operator<<` (`bool` \_\_n)
- `__ostream_type` & `operator<<` (`short` \_\_n)
- `__ostream_type` & `operator<<` (`unsigned short` \_\_n)
- `__ostream_type` & `operator<<` (`int` \_\_n)
- `__ostream_type` & `operator<<` (`unsigned int` \_\_n)
- `__ostream_type` & `operator<<` (`long long` \_\_n)
- `__ostream_type` & `operator<<` (`unsigned long long` \_\_n)
- `__ostream_type` & `operator<<` (`double` \_\_f)
- `__ostream_type` & `operator<<` (`float` \_\_f)
- `__ostream_type` & `operator<<` (`long double` \_\_f)
- `__ostream_type` & `operator<<` (`const void *` \_\_p)
- `__ostream_type` & `operator<<` (`__streambuf_type` \*\_\_sb)

### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has

several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type & put (char_type __c)`
- `void _M_write (const char_type *__s, streamsize __n)`
- `__ostream_type & write (const char_type *__s, streamsize __n)`

#### Static Public Member Functions

- static bool `sync_with_stdio` (bool \_\_sync=true)
- static int `xalloc` () throw ()

#### Static Public Attributes

- static const `fmtflags adjustfield`
- static const `openmode app`
- static const `openmode ate`
- static const `iosstate badbit`
- static const `fmtflags basefield`
- static const `seekdir beg`
- static const `openmode binary`
- static const `fmtflags boolalpha`
- static const `seekdir cur`
- static const `fmtflags dec`
- static const `seekdir end`
- static const `iosstate eofbit`
- static const `iosstate failbit`
- static const `fmtflags fixed`
- static const `fmtflags floatfield`
- static const `iosstate goodbit`
- static const `fmtflags hex`
- static const `openmode in`
- static const `fmtflags internal`
- static const `fmtflags left`
- static const `fmtflags oct`
- static const `openmode out`
- static const `fmtflags right`
- static const `fmtflags scientific`

- static const `fmtflags showbase`
- static const `fmtflags showpoint`
- static const `fmtflags showpos`
- static const `fmtflags skipws`
- static const `openmode trunc`
- static const `fmtflags unitbuf`
- static const `fmtflags uppercase`

### Protected Types

- enum { `_S_local_word_size` }

### Protected Member Functions

- void `_M_cache_locale` (const `locale` &\_\_loc)
- void `_M_call_callbacks` (`event` \_\_ev) throw ()
- void `_M_dispose_callbacks` (void) throw ()
- template<typename \_ValueT >  
  `__istream_type` & `_M_extract` (\_ValueT &\_\_v)
- `_Words` & `_M_grow_words` (int \_\_index, bool \_\_iword)
- void `_M_init` () throw ()
- template<typename \_ValueT >  
  `__ostream_type` & `_M_insert` (\_ValueT \_\_v)
- void `init` (`basic_streambuf`<\_CharT, \_Traits> \* \_\_sb)

### Protected Attributes

- `_Callback_list` \* `_M_callbacks`
- const `__ctype_type` \* `_M_ctype`
- `iostate` `_M_exception`
- `char_type` `_M_fill`
- bool `_M_fill_init`
- `fmtflags` `_M_flags`
- `streamsize` `_M_gcount`
- `locale` `_M_ios_locale`
- `_Words` `_M_local_word` [`_S_local_word_size`]
- const `__num_get_type` \* `_M_num_get`
- const `__num_put_type` \* `_M_num_put`
- `streamsize` `_M_precision`
- `basic_streambuf`<\_CharT, \_Traits> \* `_M_streambuf`
- `iostate` `_M_streambuf_state`
- `basic_ostream`<\_CharT, \_Traits> \* `_M_tie`

- [streamsize](#) `_M_width`
- `_Words * _M_word`
- `int _M_word_size`
- `_Words _M_word_zero`

## Friends

- class `sentry`
- class `sentry`

### 5.380.1 Detailed Description

`template<typename _CharT, typename _Traits> class std::basic_iostream< _CharT, _Traits >`

Merging istream and ostream capabilities.

This class multiply inherits from the input and output stream classes simply to provide a single interface.

Definition at line 771 of file `istream`.

### 5.380.2 Member Typedef Documentation

**5.380.2.1** `template<typename _CharT, typename _Traits> typedef ctype<_CharT> std::basic_istream< _CharT, _Traits >::__ctype_type [inherited]`

These are non-standard types.

Reimplemented from `std::basic_ios< _CharT, _Traits >`.

Definition at line 73 of file `istream`.

**5.380.2.2** `template<typename _CharT, typename _Traits> typedef ctype<_CharT> std::basic_ostream< _CharT, _Traits >::__ctype_type [inherited]`

These are non-standard types.

Reimplemented from `std::basic_ios< _CharT, _Traits >`.

Definition at line 73 of file `ostream`.

**5.380.2.3** `template<typename _CharT, typename _Traits> typedef  
num_get<_CharT, istreambuf_iterator<_CharT, _Traits>  
> std::basic_istream< _CharT, _Traits >::__num_get_type  
[inherited]`

These are non-standard types.

Reimplemented from `std::basic_ios< _CharT, _Traits >`.

Definition at line 72 of file `istream`.

**5.380.2.4** `template<typename _CharT, typename _Traits> typedef  
num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>  
> std::basic_ostream< _CharT, _Traits >::__num_put_type  
[inherited]`

These are non-standard types.

Reimplemented from `std::basic_ios< _CharT, _Traits >`.

Definition at line 72 of file `ostream`.

**5.380.2.5** `template<typename _CharT, typename _Traits> typedef  
num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>  
> std::basic_ios< _CharT, _Traits >::__num_put_type  
[inherited]`

These are non-standard types.

Reimplemented in `std::basic_ostream< _CharT, _Traits >`, `std::basic_ostream< char, _Traits >`, and `std::basic_ostream< char >`.

Definition at line 86 of file `basic_ios.h`.

**5.380.2.6** `template<typename _CharT, typename _Traits> typedef _CharT  
std::basic_istream< _CharT, _Traits >::char_type [inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios< _CharT, _Traits >`.

Reimplemented in `std::basic_ifstream< _CharT, _Traits >`, and `std::basic_istreamstream< _CharT, _Traits, _Alloc >`.

Definition at line 61 of file `istream`.

**5.380.2.7** `template<typename _CharT, typename _Traits> typedef _CharT  
std::basic_istream<_CharT, _Traits>::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ostream<_CharT, _Traits>`.

Reimplemented in `std::basic_fstream<_CharT, _Traits>`, and `std::basic_stringstream<_CharT, _Traits, _Alloc>`.

Definition at line 779 of file `istream`.

**5.380.2.8** `typedef void(* std::ios_base::event_callback)(event, ios_base &, int)  
[inherited]`

The type of an event callback function.

**Parameters**

*event* One of the members of the event enum.

*ios\_base* Reference to the `ios_base` object.

*int* The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 438 of file `ios_base.h`.

**5.380.2.9** `typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]`

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`

- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 257 of file `ios_base.h`.

**5.380.2.10** `template<typename _CharT, typename _Traits> typedef`  
`_Traits::int_type std::basic_istream<_CharT, _Traits>::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_ostream<\\_CharT, \\_Traits>](#).

Reimplemented in [std::basic\\_fstream<\\_CharT, \\_Traits>](#), and [std::basic\\_stringstream<\\_CharT, \\_Traits, \\_Alloc>](#).

Definition at line 780 of file `istream`.

**5.380.2.11** `template<typename _CharT, typename _Traits> typedef`  
`_Traits::int_type std::basic_istream<_CharT, _Traits>::int_type`  
`[inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.



Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Reimplemented in `std::basic_ifstream<_CharT, _Traits>`, and `std::basic_istream<_CharT, _Traits, _Alloc>`.

Definition at line 62 of file `istream`.

#### 5.380.2.12 `typedef _Ios_Iostate std::ios_base::iostate` [inherited]

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 332 of file `ios_base.h`.

#### 5.380.2.13 `template<typename _CharT, typename _Traits> typedef _Traits::off_type std::basic_ostream<_CharT, _Traits>::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ostream<_CharT, _Traits>`.

Reimplemented in `std::basic_fstream<_CharT, _Traits>`, and `std::basic_stringstream<_CharT, _Traits, _Alloc>`.

Definition at line 782 of file `istream`.

#### 5.380.2.14 `template<typename _CharT, typename _Traits> typedef _Traits::off_type std::basic_istream<_CharT, _Traits>::off_type` [inherited]

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Reimplemented in `std::basic_ifstream<_CharT, _Traits>`, and `std::basic_istream<_CharT, _Traits, _Alloc>`.

Definition at line 64 of file `istream`.

#### **5.380.2.15** `typedef _Ios_Openmode std::ios_base::openmode` [inherited]

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 363 of file `ios_base.h`.

#### **5.380.2.16** `template<typename _CharT, typename _Traits> typedef _Traits::pos_type std::basic_iostream<_CharT, _Traits>::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ostream<_CharT, _Traits>`.

Reimplemented in `std::basic_fstream<_CharT, _Traits>`, and `std::basic_stringstream<_CharT, _Traits, _Alloc>`.

Definition at line 781 of file `istream`.

**5.380.2.17** `template<typename _CharT, typename _Traits> typedef  
_Traits::pos_type std::basic_istream<_CharT, _Traits>::pos_type  
[inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Reimplemented in `std::basic_ifstream<_CharT, _Traits>`, and `std::basic_istream<_CharT, _Traits, _Alloc>`.

Definition at line 63 of file `istream`.

**5.380.2.18** `typedef _Ios_Seekdir std::ios_base::seekdir [inherited]`

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 395 of file `ios_base.h`.

**5.380.2.19** `template<typename _CharT, typename _Traits> typedef  
_Traits std::basic_istream<_CharT, _Traits>::traits_type  
[inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Reimplemented in `std::basic_ifstream<_CharT, _Traits>`, and `std::basic_istream<_CharT, _Traits, _Alloc>`.

Definition at line 65 of file `istream`.

**5.380.2.20 template<typename \_CharT, typename \_Traits> typedef \_Traits  
std::basic\_iostream< \_CharT, \_Traits >::traits\_type**

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_ostream< \\_CharT, \\_Traits >](#).

Reimplemented in [std::basic\\_fstream< \\_CharT, \\_Traits >](#), and [std::basic\\_stringstream< \\_CharT, \\_Traits, \\_Alloc >](#).

Definition at line 783 of file istream.

**5.380.3 Member Enumeration Documentation**

**5.380.3.1 enum std::ios\_base::event [inherited]**

The set of events that may be passed to an event callback.

erase\_event is used during ~ios() and copyfmt(). imbue\_event is used during [imbue\(\)](#). copyfmt\_event is used during copyfmt().

Definition at line 421 of file ios\_base.h.

**5.380.4 Constructor & Destructor Documentation**

**5.380.4.1 template<typename \_CharT, typename \_Traits>  
std::basic\_iostream< \_CharT, \_Traits >::basic\_iostream (   
basic\_streambuf< \_CharT, \_Traits > \* \_\_sb ) [inline,  
explicit]**

Constructor does nothing.

Both of the parent classes are initialized with the same streambuf pointer passed to this constructor.

Definition at line 796 of file istream.

**5.380.4.2 template<typename \_CharT, typename \_Traits> virtual  
std::basic\_iostream< \_CharT, \_Traits >::~~basic\_iostream ( )  
[inline, virtual]**

Destructor does nothing.

Definition at line 803 of file istream.

### **5.380.5 Member Function Documentation**

#### **5.380.5.1 const locale& std::ios\_base::\_M\_getloc ( ) const [inline, inherited]**

Locale access.

##### **Returns**

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 708 of file ios\_base.h.

Referenced by std::money\_get<\_CharT, \_InIter>::do\_get(), std::num\_get<\_CharT, \_InIter>::do\_get(), std::time\_get<\_CharT, \_InIter>::do\_get\_date(), std::time\_get<\_CharT, \_InIter>::do\_get\_monthname(), std::time\_get<\_CharT, \_InIter>::do\_get\_time(), std::time\_get<\_CharT, \_InIter>::do\_get\_weekday(), std::time\_get<\_CharT, \_InIter>::do\_get\_year(), std::time\_put<\_CharT, \_OutIter>::do\_put(), std::num\_put<\_CharT, \_OutIter>::do\_put(), and std::time\_put<\_CharT, \_OutIter>::put().

#### **5.380.5.2 template<typename \_CharT, typename \_Traits> void std::basic\_ostream<\_CharT, \_Traits>::\_M\_write ( const char\_type \* \_\_s, streamsize \_\_n ) [inline, inherited]**

Simple insertion.

##### **Parameters**

*c* The character to insert.

##### **Returns**

\*this

Tries to insert *c*.

##### **Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 289 of file ostream.

Referenced by `std::basic_ostream<_CharT, _Traits>::write()`.

**5.380.5.3** `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::bad ( ) const` [**inline, inherited**]

Fast error checking.

#### Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 203 of file basic\_ios.h.

**5.380.5.4** `template<typename _CharT, typename _Traits> void std::basic_ios<_CharT, _Traits>::clear ( iostate __state = goodbit )` [**inherited**]

[Re]sets the error state.

#### Parameters

*state* The new state flag(s) to set.

See [std::ios\\_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

Definition at line 42 of file basic\_ios.tcc.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::exceptions()`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::rdstate()`.

Referenced by `std::basic_ios<char, _Traits>::exceptions()`, `std::basic_fstream<_CharT, _Traits>::open()`, `std::basic_ofstream<_CharT, _Traits>::open()`, `std::basic_ifstream<_CharT, _Traits>::open()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ios<char, _Traits>::setstate()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

**5.380.5.5** `template<typename _CharT, typename _Traits > basic_ios<  
_CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (  
const basic_ios< _CharT, _Traits > & __rhs ) [inherited]`

Copies fields of \_\_rhs into this.

#### Parameters

`__rhs` The source values for the copies.

#### Returns

Reference to this object.

All fields of \_\_rhs are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy `exceptions()`.

Definition at line 64 of file `basic_ios.tcc`.

References `std::basic_ios< _CharT, _Traits >::exceptions()`, `std::basic_ios< _CharT, _Traits >::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios< _CharT, _Traits >::tie()`, and `std::ios_base::width()`.

**5.380.5.6** `template<typename _CharT, typename _Traits> bool std::basic_ios<  
_CharT, _Traits >::eof ( ) const [inline, inherited]`

Fast error checking.

#### Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 182 of file `basic_ios.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

**5.380.5.7** `template<typename _CharT, typename _Traits> void std::basic_ios<_CharT, _Traits>::exceptions ( iostate __except ) [inline, inherited]`

Throwing exceptions on errors.

#### Parameters

*except* The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
 std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

 std::ifstream f ("/etc/motd");

 std::cerr << "Setting badbit\n";
 f.setstate (std::ios_base::badbit);

 std::cerr << "Setting exception mask\n";
 f.exceptions (std::ios_base::badbit);
}
```

Definition at line 249 of file `basic_ios.h`.

**5.380.5.8** `template<typename _CharT, typename _Traits> iostate std::basic_ios<_CharT, _Traits>::exceptions ( ) const [inline, inherited]`

Throwing exceptions on errors.

#### Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(iostate)` for the meaning of the return value.



Definition at line 214 of file basic\_ios.h.

Referenced by std::basic\_ios< \_CharT, \_Traits >::clear(), and std::basic\_ios< \_CharT, \_Traits >::copyfmt().

#### **5.380.5.9 template<typename \_CharT, typename \_Traits> bool std::basic\_ios< \_CharT, \_Traits >::fail ( ) const [inline, inherited]**

Fast error checking.

##### **Returns**

True if either the badbit or the failbit is set.

Checking the badbit in [fail\(\)](#) is historical practice. Note that other iostate flags may also be set.

Definition at line 193 of file basic\_ios.h.

Referenced by std::basic\_ios< char, \_Traits >::operator void \*(), std::basic\_ios< char, \_Traits >::operator!(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_ostream< \_CharT, \_Traits >::tellp(), and std::regex\_traits< \_Ch\_type >::value().

#### **5.380.5.10 template<typename \_CharT, typename \_Traits> char\_type std::basic\_ios< \_CharT, \_Traits >::fill ( char\_type \_\_ch ) [inline, inherited]**

Sets a new *empty* character.

##### **Parameters**

*ch* The new character.

##### **Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space ( ' ' ) in the current locale.

Definition at line 382 of file basic\_ios.h.

**5.380.5.11** `template<typename _CharT, typename _Traits> char_type  
std::basic_ios<_CharT, _Traits>::fill ( ) const [inline,  
inherited]`

Retrieves the *empty* character.

**Returns**

The current fill character.

It defaults to a space ( ' ' ) in the current locale.

Definition at line 362 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, and `std::basic_ios<char, _Traits>::fill()`.

**5.380.5.12** `fmtflags std::ios_base::flags ( ) const [inline, inherited]`

Access to format flags.

**Returns**

The format control flags for both input and output.

Definition at line 553 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::operator<<()`, `std::operator>>()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

**5.380.5.13** `fmtflags std::ios_base::flags ( fmtflags __fmtfl ) [inline,  
inherited]`

Setting new format flags all at once.

**Parameters**

*fmtfl* The new flags to set.

**Returns**

The previous format control flags.

## 5.380 std::basic\_iostream<\_CharT, \_Traits> Class Template Reference 1840

---

This function overwrites all the format flags with *fmtfl*.

Definition at line 564 of file ios\_base.h.

**5.380.5.14** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::flush ( ) [inherited]`

Synchronizing the stream buffer.

### Returns

\*this

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf() -> pubsync()`, and if that returns -1, sets badbit.

Definition at line 213 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

Referenced by `std::flush()`.

**5.380.5.15** `template<typename _CharT, typename _Traits> streamsize std::basic_istream<_CharT, _Traits>::gcount ( ) const [inline, inherited]`

Character counting.

### Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 251 of file istream.

**5.380.5.16** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits>::get ( void ) [inherited]`

Simple extraction.

**Returns**

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 238 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.380.5.17** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get( char_type * __s, streamsize __n, char_type __delim )`  
[`inherited`]

Simple multiple-character extraction.

**Parameters**

- s* Pointer to an array.
- n* Maximum number of characters to store in *s*.
- delim* A "stop" character.

**Returns**

\*this

Characters are extracted and stored into *s* until one of the following happens:

- *n*−1 characters are stored
- the input sequence reaches EOF
- the next character equals *delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

**Note**

This function is not overloaded on signed char and unsigned char.

## 5.380 `std::basic_istream<_CharT, _Traits>` Class Template Reference 1842

Definition at line 311 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

**5.380.5.18** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get (char_type & __c) [inherited]`

Simple extraction.

### Parameters

*c* The character in which to store data.

### Returns

\*this

Tries to extract a character and store it in *c*. If none are available, sets failbit and returns `traits::eof()`.

### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 274 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.380.5.19** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::get (char_type * __s, streamsize __n) [inline, inherited]`

Simple multiple-character extraction.

### Parameters

*s* Pointer to an array.

*n* Maximum number of characters to store in *s*.

#### Returns

\*this

Returns `get(s,n,widen('\n'))`.

Definition at line 335 of file `istream`.

**5.380.5.20** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get ( __streambuf_type & __sb, char_type __delim ) [inherited]`

Extraction into another streambuf.

#### Parameters

*sb* A streambuf in which to store data.

*delim* A "stop" character.

#### Returns

\*this

Characters are extracted and inserted into *sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, `failbit` is set in the stream's error state.

Definition at line 358 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, `std::basic_streambuf<_CharT, _Traits>::snextc()`, and `std::basic_streambuf<_CharT, _Traits>::sputc()`.

**5.380.5.21** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::get ( __streambuf_type & __sb ) [inline, inherited]`

Extraction into another streambuf.

#### Parameters

*sb* A streambuf in which to store data.

#### Returns

\*this

Returns `get(sb, widen('\n'))`.

Definition at line 368 of file `istream`.

**5.380.5.22** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::getline ( char_type * __s, streamsize __n, char_type __delim ) [inherited]`

String extraction.

#### Parameters

*s* A character array in which to store the data.

*n* Maximum number of characters to extract.

*delim* A "stop" character.

#### Returns

\*this

Extracts and stores characters into *s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals *delim*, in which case the character is extracted (and therefore counted in `gcount()`) but not stored

3.  $n-1$  characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 402 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sbumpc()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snexctc()`.

**5.380.5.23** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::getline ( char_type * __s, streamsize __n ) [inline, inherited]`

String extraction.

#### Parameters

- s* A character array in which to store the data.
- n* Maximum number of characters to extract.

#### Returns

\*this

Returns `getline(s,n,widen('\n'))`.

Definition at line 408 of file istream.

Referenced by `std::basic_istream< char >::getline()`.

**5.380.5.24** `locale std::ios_base::getloc ( ) const [inline, inherited]`

Locale access.

#### Returns

A copy of the current locale.



## 5.380 `std::basic_iostream<_CharT, _Traits>` Class Template Reference 1846

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 697 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::money_put<_CharT, _Outputter>::do_put()`, `std::basic_ios<_CharT, _Traits>::imbue()`, `std::operator>>()`, and `std::ws()`.

**5.380.5.25** `template<typename _CharT, typename _Traits> bool  
std::basic_ios<_CharT, _Traits>::good( ) const [inline,  
inherited]`

Fast error checking.

### Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 172 of file `basic_ios.h`.

Referenced by `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

**5.380.5.26** `template<typename _CharT, typename _Traits> basic_istream<  
_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore  
( void ) [inherited]`

Discarding characters.

### Parameters

*n* Number of characters to discard.

*delim* A "stop" character.

### Returns

`*this`

Extracts characters and throws them away until one of the following happens:

- if `n != std::numeric_limits<int>::max()`, `n` characters are extracted

- the input sequence reaches end-of-file
- the next character equals *delim* (in this case, the character is extracted); note that this condition will never occur if *delim* equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 462 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sbumpc()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.380.5.27** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore( streamsize __n, int_type __delim ) [inherited]`

Simple extraction.

#### Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 557 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sbumpc()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

**5.380.5.28** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore( streamsize __n ) [inherited]`

Simple extraction.

#### Returns

A character, or `eof()`.

## 5.380 `std::basic_istream<_CharT, _Traits>` Class Template Reference 1848

---

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 495 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

### 5.380.5.29 `template<typename _CharT, typename _Traits> locale std::basic_ios<_CharT, _Traits>::imbue ( const locale & __loc ) [inherited]`

Moves to a new locale.

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Reimplemented from `std::ios_base`.

Definition at line 115 of file `basic_ios.tcc`.

References `std::ios_base::getloc()`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

Referenced by `std::operator<<()`.

### 5.380.5.30 `template<typename _CharT, typename _Traits> void std::basic_ios<_CharT, _Traits>::init ( basic_streambuf< _CharT, _Traits> * __sb ) [protected, inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 127 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

## 5.380 `std::basic_iostream<_CharT, _Traits>` Class Template Reference 1849

Referenced by `std::basic_fstream<_CharT, _Traits>::basic_fstream()`, `std::basic_ifstream<_CharT, _Traits>::basic_ifstream()`, `std::basic_ios<char, _Traits>::basic_ios()`, `std::basic_istream<char>::basic_istream()`, `std::basic_istreamstream<_CharT, _Traits, _Alloc>::basic_istreamstream()`, `std::basic_ofstream<_CharT, _Traits>::basic_ofstream()`, `std::basic_ostream<char>::basic_ostream()`, `std::basic_ostringstream<_CharT, _Traits, _Alloc>::basic_ostringstream()`, and `std::basic_stringstream<_CharT, _Traits, _Alloc>::basic_stringstream()`.

### 5.380.5.31 `long& std::ios_base::iword ( int __ix ) [inline, inherited]`

Access to integer array.

#### Parameters

`__ix` Index into the array.

#### Returns

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 743 of file `ios_base.h`.

### 5.380.5.32 `template<typename _CharT, typename _Traits> char std::basic_ios<_CharT, _Traits>::narrow ( char_type __c, char __dfault ) const [inline, inherited]`

Squeezes characters.

#### Parameters

`c` The character to narrow.

`dfault` The character to narrow.

#### Returns

The narrowed character.

## 5.380 `std::basic_iostream<_CharT, _Traits>` Class Template Reference 1850

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, default)
```

Additional notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 422 of file `basic_ios.h`.

**5.380.5.33** `template<typename _CharT, typename _Traits> std::basic_ios<_CharT, _Traits>::operator void * ( ) const` [`inline`, `inherited`]

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 113 of file `basic_ios.h`.

**5.380.5.34** `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::operator! ( ) const` [`inline`, `inherited`]

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 117 of file `basic_ios.h`.

**5.380.5.35** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< ( bool __n )` [`inline`, `inherited`]

Basic arithmetic inserters.

### Parameters

*A* variable of builtin type.

### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 175 of file `ostream`.

**5.380.5.36** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream<_CharT, _Traits>::operator<<( unsigned  
long __n ) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 171 of file `ostream`.

**5.380.5.37** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream<_CharT, _Traits>::operator<<( long __n )  
[inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 167 of file `ostream`.

**5.380.5.38** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream<_CharT, _Traits>::operator<< ( ios_base  
&(*)(ios_base &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the `iomanip` header.

Definition at line 129 of file `ostream`.

**5.380.5.39** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream<_CharT, _Traits>::operator<< (   
__ostream_type &(*)(__ostream_type &) __pf ) [inline,  
inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the `iomanip` header.

Definition at line 110 of file `ostream`.

**5.380.5.40** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream<_CharT, _Traits>::operator<< ( __ios_type  
&(*)(__ios_type &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the `iomanip` header.

Definition at line 119 of file `ostream`.

**5.380.5.41** `template<typename _CharT, typename _Traits> basic_ostream<  
_CharT, _Traits> & std::basic_ostream<_CharT, _Traits  
>::operator<< ( short __n ) [inherited]`

Basic arithmetic inserters.

## 5.380 `std::basic_iostream<_CharT, _Traits>` Class Template Reference 1853

---

### Parameters

*A* variable of builtin type.

### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 94 of file `ostream.tcc`.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

**5.380.5.42** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< ( unsigned short __n ) [inline, inherited]`

Basic arithmetic inserters.

### Parameters

*A* variable of builtin type.

### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 182 of file `ostream`.

**5.380.5.43** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<< ( int __n ) [inherited]`

Basic arithmetic inserters.

### Parameters

*A* variable of builtin type.



**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 108 of file `ostream.tcc`.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

**5.380.5.44** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream<_CharT, _Traits>::operator<< ( long long  
__n ) [inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 202 of file `ostream`.

**5.380.5.45** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream<_CharT, _Traits>::operator<< ( double __f  
) [inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 211 of file `ostream`.

**5.380.5.46** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream<_CharT, _Traits>::operator<<( long double  
__f ) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 223 of file `ostream`.

**5.380.5.47** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream<_CharT, _Traits>::operator<<( float __f )  
[inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 215 of file `ostream`.

**5.380.5.48** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( const void * __p ) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 227 of file ostream.

**5.380.5.49** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<<( __streambuf_type * __sb ) [inherited]`

Extracting from another streambuf.

#### Parameters

*sb* A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from *sb* and inserted into \*this until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from *sb*, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 122 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.380.5.50** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( unsigned int __n ) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 193 of file ostream.

**5.380.5.51** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( unsigned long long __n ) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 206 of file ostream.

**5.380.5.52** `template<typename _CharT, typename _Traits> __istream_type&  
std::basic_istream<_CharT, _Traits>::operator>> ( void *& __p  
) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 217 of file `istream`.

**5.380.5.53** `template<typename _CharT, typename _Traits> __istream_type&  
std::basic_istream<_CharT, _Traits>::operator>> (   
__istream_type &(*)(__istream_type &) __pf ) [inline,  
inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 122 of file `istream`.

**5.380.5.54** `template<typename _CharT, typename _Traits> __istream_type&  
std::basic_istream<_CharT, _Traits>::operator>> ( unsigned int  
& __n ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 183 of file `istream`.

**5.380.5.55** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::operator>> ( __streambuf_type * __sb ) [inherited]`

Extracting into another streambuf.

#### Parameters

*sb* A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the *sb* streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 206 of file `istream.tcc`.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.380.5.56** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( unsigned long long & __n ) [inline, inherited]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 200 of file `istream`.

**5.380.5.57** `template<typename _CharT, typename _Traits> __istream_type&  
std::basic_istream<_CharT, _Traits>::operator>> ( __ios_type  
&(*)(__ios_type &) pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iomanip` header.

Definition at line 126 of file `istream`.

**5.380.5.58** `template<typename _CharT, typename _Traits> __istream_type&  
std::basic_istream<_CharT, _Traits>::operator>> ( long & n  
) [inline, inherited]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 187 of file `istream`.

**5.380.5.59** `template<typename _CharT, typename _Traits> __istream_type&  
std::basic_istream<_CharT, _Traits>::operator>> ( bool & __n  
) [inline, inherited]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 169 of file `istream`.

**5.380.5.60** `template<typename _CharT, typename _Traits> __istream_type&  
std::basic_istream<_CharT, _Traits>::operator>> ( unsigned  
long & __n ) [inline, inherited]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 191 of file `istream`.

**5.380.5.61** `template<typename _CharT, typename _Traits> basic_istream<  
_CharT, _Traits> & std::basic_istream<_CharT, _Traits  
>::operator>> ( short & __n ) [inherited]`

Basic arithmetic extractors.



### Parameters

*A* variable of builtin type.

### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 116 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get<_CharT, _InIter>::get()`, `std::ios_base::goodbit`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.380.5.62** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( long long & __n ) [inline, inherited]`

Basic arithmetic extractors.

### Parameters

*A* variable of builtin type.

### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 196 of file `istream`.

**5.380.5.63** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( ios_base &(*)(ios_base &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 133 of file `istream`.

**5.380.5.64** `template<typename _CharT, typename _Traits> __istream_type&  
std::basic_istream<_CharT, _Traits>::operator>> ( double &  
__f ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

`A` variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 209 of file `istream`.

**5.380.5.65** `template<typename _CharT, typename _Traits> __istream_type&  
std::basic_istream<_CharT, _Traits>::operator>> ( float & __f  
) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

`A` variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 205 of file `istream`.

**5.380.5.66** `template<typename _CharT, typename _Traits> basic_istream<  
_CharT, _Traits> & std::basic_istream<_CharT, _Traits  
>::operator>> ( int & __n ) [inherited]`

Basic arithmetic extractors.

## 5.380 `std::basic_istream<_CharT, _Traits>` Class Template Reference 1864

### Parameters

*A* variable of builtin type.

### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 161 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get<_CharT, _InIter>::get()`, `std::ios_base::goodbit`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.380.5.67** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( long double & __f ) [inline, inherited]`

Basic arithmetic extractors.

### Parameters

*A* variable of builtin type.

### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 213 of file `istream`.

**5.380.5.68** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( unsigned short & __n ) [inline, inherited]`

Basic arithmetic extractors.

### Parameters

*A* variable of builtin type.

### Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 176 of file `istream`.

**5.380.5.69** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits>::peek( void ) [inherited]`

Looking ahead in the stream.

### Returns

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 622 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.380.5.70** `streamsize std::ios_base::precision( streamsize __prec ) [inline, inherited]`

Changing flags.

### Parameters

*prec* The new precision value.

### Returns

The previous value of `precision()`.

Definition at line 632 of file `ios_base.h`.

**5.380.5.71** streamsize std::ios\_base::precision ( ) const [inline, inherited]

Flags access.

#### Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 623 of file ios\_base.h.

Referenced by std::basic\_ios<\_CharT, \_Traits>::copyfmt(), and std::operator<<().

**5.380.5.72** template<typename \_CharT, typename \_Traits> basic\_ostream<\_CharT, \_Traits> & std::basic\_ostream<\_CharT, \_Traits>::put ( char\_type \_\_c ) [inherited]

Simple insertion.

#### Parameters

*c* The character to insert.

#### Returns

\*this

Tries to insert *c*.

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 151 of file ostream.tcc.

References std::ios\_base::badbit, std::ios\_base::goodbit, std::basic\_ios<\_CharT, \_Traits>::rdbuf(), and std::basic\_ios<\_CharT, \_Traits>::setstate().

Referenced by std::endl(), and std::ends().

**5.380.5.73** template<typename \_CharT, typename \_Traits> basic\_istream<\_CharT, \_Traits> & std::basic\_istream<\_CharT, \_Traits>::putback ( char\_type \_\_c ) [inherited]

Unextracting a single character.

#### Parameters

`c` The character to push back into the input stream.

#### Returns

`*this`

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

#### Note

Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 713 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::clear()`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::rdstate()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sputbackc()`.

Referenced by `std::operator>>()`.

#### 5.380.5.74 `void*& std::ios_base::pword ( int __ix ) [inline, inherited]`

Access to void pointer array.

#### Parameters

`__ix` Index into the array.

#### Returns

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 764 of file `ios_base.h`.

```
5.380.5.75 template<typename _CharT, typename _Traits>
 basic_streambuf<_CharT, _Traits>* std::basic_ios<_CharT,
 _Traits>::rdbuf() const [inline, inherited]
```

Accessing the underlying buffer.

#### Returns

The current stream buffer.

This does not change the state of the stream.

Reimplemented in [std::basic\\_ifstream<\\_CharT, \\_Traits>](#), [std::basic\\_ofstream<\\_CharT, \\_Traits>](#), [std::basic\\_fstream<\\_CharT, \\_Traits>](#), [std::basic\\_istream<\\_CharT, \\_Traits, \\_Alloc>](#), [std::basic\\_ostringstream<\\_CharT, \\_Traits, \\_Alloc>](#), and [std::basic\\_stringstream<\\_CharT, \\_Traits, \\_Alloc>](#).

Definition at line 313 of file `basic_ios.h`.

Referenced by `std::basic_ostream<char>::_M_write()`, `std::basic_ios<_CharT, _Traits>::clear()`, `std::basic_ostream<_CharT, _Traits>::flush()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_ios<_CharT, _Traits>::imbue()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_ostream<_CharT, _Traits>::put()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::readsome()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, `std::basic_istream<_CharT, _Traits>::sync()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_ostream<_CharT, _Traits>::tellp()`, `std::basic_istream<_CharT, _Traits>::unset()`, and `std::ws()`.

```
5.380.5.76 template<typename _CharT, typename _Traits> basic_streambuf<
 _CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rdbuf (
 basic_streambuf<_CharT, _Traits> * __sb) [inherited]
```

Changing the underlying buffer.

### Parameters

*sb* The new stream buffer.

### Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream foo; // or some other derived type
std::streambuf* p =;

foo.ios::rdbuf(p); // ios == basic_ios<char>
```

Definition at line 54 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::clear()`.

#### 5.380.5.77 `template<typename _CharT, typename _Traits> iostate std::basic_ios<_CharT, _Traits>::rdstate( ) const [inline, inherited]`

Returns the error state of the stream buffer.

### Returns

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 129 of file `basic_ios.h`.

Referenced by `std::basic_ios<char, _Traits>::bad()`, `std::basic_ios<_CharT, _Traits>::clear()`, `std::basic_ios<char, _Traits>::eof()`, `std::basic_ios<char, _Traits>::fail()`, `std::basic_ios<char, _Traits>::good()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ios<char, _Traits>::setstate()`, and `std::basic_istream<_CharT, _Traits>::unget()`.



**5.380.5.78** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::read (char_type * __s, streamsize __n) [inherited]`

Extraction without delimiters.

#### Parameters

- s* A character array.
- n* Maximum number of characters to store.

#### Returns

\*this

If the stream state is `good()`, extracts characters and stores them into *s* until one of the following happens:

- *n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 652 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.380.5.79** `template<typename _CharT, typename _Traits> streamsize std::basic_istream<_CharT, _Traits>::readsome (char_type * __s, streamsize __n) [inherited]`

Extraction until the buffer is exhausted, but no more.

#### Parameters

- s* A character array.
- n* Maximum number of characters to store.

**Returns**

The number of characters extracted.

Extracts characters and stores them into *s* depending on the number of characters remaining in the streambuf's buffer, `rddbuf() -> in_avail()`, called *A* here:

- if *A* == -1, sets eofbit and extracts no characters
- if *A* == 0, extracts no characters
- if *A* > 0, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 681 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::min()`, `std::basic_ios<_CharT, _Traits>::rddbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.380.5.80** `void std::ios_base::register_callback ( event_callback __fn, int __index ) [inherited]`

Add the callback `__fn` with parameter `__index`.

**Parameters**

`__fn` The function to add.

`__index` The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**5.380.5.81** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg ( pos_type __pos ) [inherited]`

Changing the current read position.

**Parameters**

*pos* A file position object.

**Returns**

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 847 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::clear()`, `std::ios_base::eofbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::rdstate()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.380.5.82** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg( off_type __off, ios_base::seekdir __dir ) [inherited]`

Changing the current read position.

**Parameters**

*off* A file offset object.

*dir* The direction in which to seek.

**Returns**

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 886 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::clear()`, `std::ios_base::eofbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`,

## 5.380 `std::basic_iostream<_CharT, _Traits>` Class Template Reference 1873

`std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::rdstate()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.380.5.83** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp ( off_type __off, ios_base::seekdir __dir ) [inherited]`

Changing the current write position.

### Parameters

*off* A file offset object.

*dir* The direction in which to seek.

### Returns

\*this

If `fail()` is not true, calls `rdbuf() -> pubseekoff (off, dir)`. If that function fails, sets failbit.

Definition at line 292 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.380.5.84** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp ( pos_type __pos ) [inherited]`

Changing the current write position.

### Parameters

*pos* A file position object.

### Returns

\*this

If `fail()` is not true, calls `rdbuf() -> pubseekpos (pos)`. If that function fails, sets failbit.

Definition at line 260 of file ostream.tcc.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.380.5.85** `fmtflags std::ios_base::setf ( fmtflags __fmtfl, fmtflags __mask )`  
`[inline, inherited]`

Setting new format flags.

#### Parameters

*fmtfl* Additional flags to set.  
*mask* The flags mask for *fmtfl*.

#### Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 597 of file ios\_base.h.

**5.380.5.86** `fmtflags std::ios_base::setf ( fmtflags __fmtfl )` `[inline, inherited]`

Setting new format flags.

#### Parameters

*fmtfl* Additional flags to set.

#### Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 580 of file ios\_base.h.

Referenced by `std::dec()`, `std::fixed()`, `std::hex()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

**5.380.5.87** `template<typename _CharT, typename _Traits> void  
std::basic_ios<_CharT, _Traits>::setstate ( iostate __state )  
[inline, inherited]`

Sets additional flags in the error state.

#### Parameters

*state* The additional state flag(s) to set.

See [std::ios\\_base::iostate](#) for the possible bit values.

Definition at line 149 of file `basic_ios.h`.

Referenced by `std::basic_ostream<char>::_M_write()`, `std::basic_fstream<_CharT, _Traits>::close()`, `std::basic_ofstream<_CharT, _Traits>::close()`, `std::basic_ifstream<_CharT, _Traits>::close()`, `std::basic_ostream<_CharT, _Traits>::flush()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_fstream<_CharT, _Traits>::open()`, `std::basic_ofstream<_CharT, _Traits>::open()`, `std::basic_ifstream<_CharT, _Traits>::open()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_ostream<_CharT, _Traits>::put()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::readsome()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, `std::basic_istream<_CharT, _Traits>::sync()`, `std::basic_istream<_CharT, _Traits>::unget()`, and `std::ws()`.

**5.380.5.88** `template<typename _CharT, typename _Traits> int  
std::basic_istream<_CharT, _Traits>::sync ( void )  
[inherited]`

Synchronizing the stream buffer.

#### Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

#### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 783 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_streambuf<_CharT, _Traits>::pubsync()`, `std::basic_istream<_CharT, _Traits>::rdbuf()`, and `std::basic_istream<_CharT, _Traits>::setstate()`.

**5.380.5.89** `static bool std::ios_base::sync_with_stdio ( bool __sync = true )`  
**[static, inherited]**

Interaction with the standard C I/O objects.

#### Parameters

*sync* Whether to synchronize or not.

#### Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

**5.380.5.90** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::pos_type std::basic_istream<_CharT, _Traits>::tellg ( void )` **[inherited]**

Getting the current read position.

#### Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, in)`.

## Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 819 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::in`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

**5.380.5.91** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>::pos_type std::basic_ostream<_CharT, _Traits>::tellp( ) [inherited]`

Getting the current write position.

## Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, out)`.

Definition at line 239 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::out`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

**5.380.5.92** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie( basic_ostream<_CharT, _Traits>* __tiestr ) [inline, inherited]`

Ties this stream to an output stream.

## Parameters

*tiestr* The output stream.

## Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 299 of file `basic_ios.h`.



**5.380.5.93** `template<typename _CharT, typename _Traits>  
 basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits  
 >::tie( ) const [inline, inherited]`

Fetches the current *tied* stream.

#### Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 287 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

**5.380.5.94** `template<typename _CharT, typename _Traits> basic_istream<  
 _CharT, _Traits> & std::basic_istream<_CharT, _Traits>::unget  
 ( void ) [inherited]`

Unextracting the previous character.

#### Returns

`*this`

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

#### Note

Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 748 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::clear()`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::rdstate()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sungetc()`.

**5.380.5.95** void std::ios\_base::unsetf ( fmtflags \_\_mask ) [inline, inherited]

Clearing format flags.

#### Parameters

*mask* The flags to unset.

This function clears *mask* in the format flags.

Definition at line 612 of file ios\_base.h.

Referenced by std::noboolalpha(), std::noshowbase(), std::noshowpoint(), std::noshowpos(), std::noskipws(), std::nounitbuf(), and std::nouppercase().

**5.380.5.96** template<typename \_CharT, typename \_Traits> char\_type  
std::basic\_ios<\_CharT, \_Traits>::widen ( char \_\_c ) const  
[inline, inherited]

Widens characters.

#### Parameters

*c* The character to widen.

#### Returns

The widened character.

Maps a character of char to a character of char\_type.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.ht>

Definition at line 441 of file basic\_ios.h.

Referenced by std::endl(), std::basic\_ios< char, \_Traits>::fill(), std::basic\_istream< char>::get(), std::basic\_istream< char>::getline(), std::getline(), and std::operator>>().

**5.380.5.97** `streamsize std::ios_base::width ( ) const [inline, inherited]`

Flags access.

#### Returns

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 646 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::operator>>()`.

**5.380.5.98** `streamsize std::ios_base::width ( streamsize __wide ) [inline, inherited]`

Changing flags.

#### Parameters

*wide* The new width value.

#### Returns

The previous value of `width()`.

Definition at line 655 of file `ios_base.h`.

**5.380.5.99** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::write ( const char_type * __s, streamsize __n ) [inherited]`

Character string insertion.

#### Parameters

*s* The array to insert.

*n* Maximum number of characters to insert.

### Returns

\*this

Characters are copied from *s* and inserted into the stream until one of the following happens:

- *n* characters are inserted
- inserting into the output sequence fails (in this case, badbit will be set in the stream's error state)

### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 185 of file ostream.tcc.

References `std::basic_ostream<_CharT, _Traits>::_M_write()`, and `std::ios_base::badbit`.

**5.380.5.100** `static int std::ios_base::xalloc ( ) throw () [static, inherited]`

Access to unique indices.

### Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

## 5.380.6 Member Data Documentation

**5.380.6.1** `template<typename _CharT, typename _Traits> streamsize std::basic_istream<_CharT, _Traits>::_M_gcount [protected, inherited]`

The number of characters extracted in the previous unformatted function; see [gcount\(\)](#).

Definition at line 81 of file istream.

Referenced by `std::basic_istream< char >::gcount()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::basic_istream< char >::~~basic_istream()`.

#### **5.380.6.2 const fmtflags std::ios\_base::adjustfield [static, inherited]**

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 312 of file ios\_base.h.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

#### **5.380.6.3 const openmode std::ios\_base::app [static, inherited]**

Seek to end before each write.

Definition at line 366 of file ios\_base.h.

#### **5.380.6.4 const openmode std::ios\_base::ate [static, inherited]**

Open and seek to end immediately after opening.

Definition at line 369 of file ios\_base.h.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

#### **5.380.6.5 const iostate std::ios\_base::badbit [static, inherited]**

Indicates a loss of integrity in an input or output sequence (such /// as an irrecoverable read error from a file).

Definition at line 336 of file ios\_base.h.

Referenced by `std::basic_ostream< char >::M_write()`, `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::basic_ios< char, _Traits >::fail()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ios< _CharT, _Traits >::init()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, `std::basic_istream< _CharT, _Traits >::unget()`, `std::basic_ostream< _CharT, _Traits >::write()`, and `std::basic_ostream< _CharT, _Traits >::sentry::~~sentry()`.

#### 5.380.6.6 `const fmtflags std::ios_base::basefield` [`static`, `inherited`]

A mask of `dec|oct|hex`. Useful for the 2-arg form of `setf`.

Definition at line 315 of file `ios_base.h`.

Referenced by `std::dec()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::hex()`, `std::oct()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

#### 5.380.6.7 `const seekdir std::ios_base::beg` [`static`, `inherited`]

Request a seek relative to the beginning of the stream.

Definition at line 398 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::seekpos()`.

#### 5.380.6.8 `const openmode std::ios_base::binary` [`static`, `inherited`]

Perform input and output in binary mode (as opposed to text mode). `/// This is probably not what you think it is; see http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html.`

Definition at line 374 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::showmanyc()`.

**5.380.6.9 const fmtflags std::ios\_base::boolalpha [static, inherited]**

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 260 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::noboolalpha()`.

**5.380.6.10 const seekdir std::ios\_base::cur [static, inherited]**

Request a seek relative to the current position within the sequence.

Definition at line 401 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::pbackfail()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_filebuf<_CharT, _Traits>::seekoff()`, `std::basic_istream<_CharT, _Traits>::tellg()`, and `std::basic_ostream<_CharT, _Traits>::tellp()`.

**5.380.6.11 const fmtflags std::ios\_base::dec [static, inherited]**

Converts integer input or generates integer output in decimal base.

Definition at line 263 of file `ios_base.h`.

Referenced by `std::dec()`.

**5.380.6.12 const seekdir std::ios\_base::end [static, inherited]**

Request a seek relative to the current end of the sequence.

Definition at line 404 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::open()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`.

**5.380.6.13 const iostate std::ios\_base::eofbit [static, inherited]**

Indicates that an input operation reached the end of an input sequence.

Definition at line 339 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_date(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::eof(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::unget(), and std::ws().

#### 5.380.6.14 const iostate std::ios\_base::failbit [static, inherited]

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 344 of file ios\_base.h.

Referenced by std::basic\_fstream< \_CharT, \_Traits >::close(), std::basic\_ofstream< \_CharT, \_Traits >::close(), std::basic\_ifstream< \_CharT, \_Traits >::close(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_date(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::fail(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_fstream< \_CharT, \_Traits >::open(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry(), and std::basic\_istream< \_CharT, \_Traits >::sentry::sentry().

#### 5.380.6.15 const fmtflags std::ios\_base::fixed [static, inherited]

Generate floating-point output in fixed-point notation.

Definition at line 266 of file ios\_base.h.



Referenced by std::fixed().

**5.380.6.16 const fmtflags std::ios\_base::floatfield [static, inherited]**

A mask of scientific|fixed. Useful for the 2-arg form of setf.

Definition at line 318 of file ios\_base.h.

Referenced by std::fixed(), and std::scientific().

**5.380.6.17 const iostate std::ios\_base::goodbit [static, inherited]**

Indicates all is well.

Definition at line 347 of file ios\_base.h.

Referenced by std::num\_get<\_CharT, \_InIter>::do\_get(), std::time\_get<\_CharT, \_InIter>::do\_get\_monthname(), std::time\_get<\_CharT, \_InIter>::do\_get\_weekday(), std::time\_get<\_CharT, \_InIter>::do\_get\_year(), std::basic\_ostream<\_CharT, \_Traits>::flush(), std::basic\_istream<\_CharT, \_Traits>::get(), std::basic\_istream<\_CharT, \_Traits>::getline(), std::basic\_istream<\_CharT, \_Traits>::ignore(), std::basic\_ios<\_CharT, \_Traits>::init(), std::basic\_ostream<\_CharT, \_Traits>::operator<<(), std::operator>>(), std::basic\_istream<\_CharT, \_Traits>::operator>>(), std::basic\_istream<\_CharT, \_Traits>::peek(), std::basic\_ostream<\_CharT, \_Traits>::put(), std::basic\_istream<\_CharT, \_Traits>::putback(), std::basic\_istream<\_CharT, \_Traits>::read(), std::basic\_istream<\_CharT, \_Traits>::readsome(), std::basic\_istream<\_CharT, \_Traits>::seekg(), std::basic\_ostream<\_CharT, \_Traits>::seekp(), std::basic\_istream<\_CharT, \_Traits>::sentry::sentry(), std::basic\_istream<\_CharT, \_Traits>::sync(), and std::basic\_istream<\_CharT, \_Traits>::unget().

**5.380.6.18 const fmtflags std::ios\_base::hex [static, inherited]**

Converts integer input or generates integer output in hexadecimal base.

Definition at line 269 of file ios\_base.h.

Referenced by std::num\_get<\_CharT, \_InIter>::do\_get(), std::num\_put<\_CharT, \_OutIter>::do\_put(), std::hex(), and std::basic\_ostream<\_CharT, \_Traits>::operator<<().

**5.380.6.19 `const openmode std::ios_base::in` [static, inherited]**

Open for input. Default for `ifstream` and `fstream`.

Definition at line 377 of file `ios_base.h`.

Referenced by `std::basic_filebuf<char_type, traits_type>::_M_set_buffer()`, `std::basic_ifstream<_CharT, _Traits>::open()`, `std::basic_filebuf<_CharT, _Traits>::_pbackfail()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::showmanyc()`, `std::basic_filebuf<_CharT, _Traits>::showmanyc()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::underflow()`, `std::basic_filebuf<_CharT, _Traits>::underflow()`, and `std::basic_filebuf<_CharT, _Traits>::xsgetn()`.

**5.380.6.20 `const fmtflags std::ios_base::internal` [static, inherited]**

Adds fill characters at a designated internal point in certain `///` generated output, or identical to `right` if no such point is `///` designated.

Definition at line 274 of file `ios_base.h`.

Referenced by `std::internal()`.

**5.380.6.21 `const fmtflags std::ios_base::left` [static, inherited]**

Adds fill characters on the right (final positions) of certain `///` generated output. (I.e., the thing you print is flush left.).

Definition at line 278 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`, and `std::left()`.

**5.380.6.22 `const fmtflags std::ios_base::oct` [static, inherited]**

Converts integer input or generates integer output in octal base.

Definition at line 281 of file `ios_base.h`.

Referenced by `std::oct()`, and `std::basic_ostream<_CharT, _Traits>::operator<<()`.

**5.380.6.23 const openmode std::ios\_base::out [static, inherited]**

Open for output. Default for `ofstream` and `fstream`.

Definition at line 380 of file `ios_base.h`.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_set_buffer()`, `std::basic_ofstream< _CharT, _Traits >::open()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

**5.380.6.24 const fmtflags std::ios\_base::right [static, inherited]**

Adds fill characters on the left (initial positions) of certain /// generated output. (I.e., the thing you print is flush right.).

Definition at line 285 of file `ios_base.h`.

Referenced by `std::right()`.

**5.380.6.25 const fmtflags std::ios\_base::scientific [static, inherited]**

Generates floating-point output in scientific notation.

Definition at line 288 of file `ios_base.h`.

Referenced by `std::scientific()`.

**5.380.6.26 const fmtflags std::ios\_base::showbase [static, inherited]**

Generates a prefix indicating the numeric base of generated integer /// output.

Definition at line 292 of file `ios_base.h`.

Referenced by `std::noshowbase()`, and `std::showbase()`.

**5.380.6.27** `const fmtflags std::ios_base::showpoint` [`static`, `inherited`]

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 296 of file `ios_base.h`.

Referenced by `std::noshowpoint()`, and `std::showpoint()`.

**5.380.6.28** `const fmtflags std::ios_base::showpos` [`static`, `inherited`]

Generates a + sign in non-negative generated numeric output.

Definition at line 299 of file `ios_base.h`.

Referenced by `std::noshowpos()`, and `std::showpos()`.

**5.380.6.29** `const fmtflags std::ios_base::skipws` [`static`, `inherited`]

Skips leading white space before certain input operations.

Definition at line 302 of file `ios_base.h`.

Referenced by `std::noskipws()`, `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, and `std::skipws()`.

**5.380.6.30** `const openmode std::ios_base::trunc` [`static`, `inherited`]

Open for input. Default for `ofstream`.

Definition at line 383 of file `ios_base.h`.

**5.380.6.31** `const fmtflags std::ios_base::unitbuf` [`static`, `inherited`]

Flushes output after each output operation.

Definition at line 305 of file `ios_base.h`.

Referenced by `std::nounitbuf()`, `std::unitbuf()`, and `std::basic_ostream<_CharT, _Traits>::sentry::~sentry()`.

5.380.6.32 `const fmtflags std::ios_base::uppercase` [static, inherited]

Replaces certain lowercase letters with their uppercase equivalents /// in generated output.

Definition at line 309 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`, `std::nouppercase()`, and `std::uppercase()`.

The documentation for this class was generated from the following file:

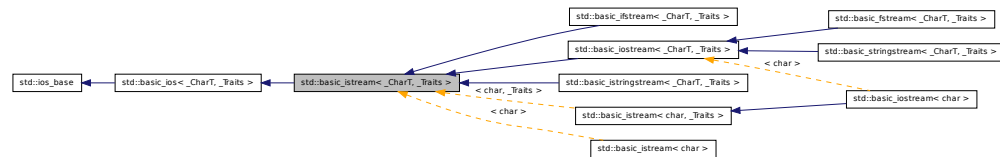
- [istream](#)

5.381 `std::basic_istream<_CharT, _Traits>` Class Template Reference

Controlling input.

This is the base class for all input streams. It provides text formatting of all builtin types, and communicates with any class derived from [basic\\_streambuf](#) to do the actual input.

Inheritance diagram for `std::basic_istream<_CharT, _Traits>`:



## Classes

- class [sentry](#)  
*Performs setup work for input streams.*

## Public Types

- typedef [ctype<\\_CharT>](#) `__ctype_type`
- typedef [basic\\_ios<\\_CharT, \\_Traits>](#) `__ios_type`

- typedef `basic_istream<_CharT, _Traits>` `__istream_type`
- typedef `num_get<_CharT, istreambuf_iterator<_CharT, _Traits>>` `__num_get_type`
- typedef `basic_streambuf<_CharT, _Traits>` `__streambuf_type`
- typedef `_CharT` `char_type`
- enum `event` { `erase_event`, `imbue_event`, `copyfmt_event` }
- typedef void(\* `event_callback`)(`event`, `ios_base` &, int)
- typedef `_Ios_Fmtflags` `fmtflags`
- typedef `_Traits::int_type` `int_type`
- typedef int `io_state`
- typedef `_Ios_Iostate` `iostate`
- typedef `_Traits::off_type` `off_type`
- typedef int `open_mode`
- typedef `_Ios_Openmode` `openmode`
- typedef `_Traits::pos_type` `pos_type`
- typedef int `seek_dir`
- typedef `_Ios_Seekdir` `seekdir`
- typedef `std::streamoff` `streamoff`
- typedef `std::streampos` `streampos`
- typedef `_Traits` `traits_type`
  
- typedef `num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>>` `__num_put_type`

### Public Member Functions

- `basic_istream` (`__streambuf_type` \*\_\_sb)
- virtual `~basic_istream` ()
- const `locale` & `_M_getloc` () const
- void `_M_setstate` (`iostate` \_\_state)
- bool `bad` () const
- void `clear` (`iostate` \_\_state=`goodbit`)
- `basic_ios` & `copyfmt` (const `basic_ios` &\_\_rhs)
- bool `eof` () const
- `iostate` `exceptions` () const
- void `exceptions` (`iostate` \_\_except)
- bool `fail` () const
- `char_type` `fill` () const
- `char_type` `fill` (`char_type` \_\_ch)
- `fmtflags` `flags` (`fmtflags` \_\_fmtfl)
- `fmtflags` `flags` () const
- `streamsize` `gcount` () const

- template<>  
basic\_istream< char > & getline (char\_type \*\_\_s, streamsize \_\_n, char\_type \_\_delim)
  - template<>  
basic\_istream< wchar\_t > & getline (char\_type \*\_\_s, streamsize \_\_n, char\_type \_\_delim)
  - locale getloc () const
  - bool good () const
  - template<>  
basic\_istream< wchar\_t > & ignore (streamsize \_\_n)
  - template<>  
basic\_istream< wchar\_t > & ignore (streamsize \_\_n, int\_type \_\_delim)
  - template<>  
basic\_istream< char > & ignore (streamsize \_\_n)
  - template<>  
basic\_istream< char > & ignore (streamsize \_\_n, int\_type \_\_delim)
  - locale imbue (const locale &\_\_loc)
  - long & iword (int \_\_ix)
  - char narrow (char\_type \_\_c, char \_\_dfault) const
  - streamsize precision (streamsize \_\_prec)
  - streamsize precision () const
  - void \*& pword (int \_\_ix)
  - basic\_streambuf< \_CharT, \_Traits > \* rdbuf (basic\_streambuf< \_CharT, \_Traits > \*\_\_sb)
  - basic\_streambuf< \_CharT, \_Traits > \* rdbuf () const
  - iostate rdstate () const
  - void register\_callback (event\_callback \_\_fn, int \_\_index)
  - fmtflags setf (fmtflags \_\_fmtfl, fmtflags \_\_mask)
  - fmtflags setf (fmtflags \_\_fmtfl)
  - void setstate (iostate \_\_state)
  - basic\_ostream< \_CharT, \_Traits > \* tie () const
  - basic\_ostream< \_CharT, \_Traits > \* tie (basic\_ostream< \_CharT, \_Traits > \*\_\_\_tiestr)
  - void unsetf (fmtflags \_\_mask)
  - char\_type widen (char \_\_c) const
  - streamsize width (streamsize \_\_wide)
  - streamsize width () const
- 
- \_\_istream\_type & operator>> (\_\_istream\_type &(\_\_pf)(\_\_istream\_type &))
  - \_\_istream\_type & operator>> (\_\_ios\_type &(\_\_pf)(\_\_ios\_type &))
  - \_\_istream\_type & operator>> (ios\_base &(\_\_pf)(ios\_base &))

### Arithmetic Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__istream_type & operator>> (bool &__n)`
- `__istream_type & operator>> (short &__n)`
- `__istream_type & operator>> (unsigned short &__n)`
- `__istream_type & operator>> (int &__n)`
- `__istream_type & operator>> (unsigned int &__n)`
- `__istream_type & operator>> (long &__n)`
- `__istream_type & operator>> (unsigned long &__n)`
- `__istream_type & operator>> (long long &__n)`
- `__istream_type & operator>> (unsigned long long &__n)`
- `__istream_type & operator>> (float &__f)`
- `__istream_type & operator>> (double &__f)`
- `__istream_type & operator>> (long double &__f)`
- `__istream_type & operator>> (void *&__p)`
- `__istream_type & operator>> (__streambuf_type *__sb)`

### Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `int_type get ()`
- `__istream_type & get (char_type &__c)`
- `__istream_type & get (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & get (char_type *__s, streamsize __n)`
- `__istream_type & get (__streambuf_type &__sb, char_type __delim)`
- `__istream_type & get (__streambuf_type &__sb)`



- `__istream_type & getline (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & getline (char_type *__s, streamsize __n)`
- `__istream_type & ignore ()`
- `__istream_type & ignore (streamsize __n)`
- `__istream_type & ignore (streamsize __n, int_type __delim)`
- `int_type peek ()`
- `__istream_type & read (char_type *__s, streamsize __n)`
- `streamsize readsome (char_type *__s, streamsize __n)`
- `__istream_type & putback (char_type __c)`
- `__istream_type & unget ()`
- `int sync ()`
- `pos_type tellg ()`
- `__istream_type & seekg (pos_type)`
- `__istream_type & seekg (off_type, ios_base::seekdir)`
- `operator void * () const`
- `bool operator! () const`

#### Static Public Member Functions

- static bool `sync_with_stdio` (bool \_\_sync=true)
- static int `xalloc` () throw ()

#### Static Public Attributes

- static const `fmtflags adjustfield`
- static const `openmode app`
- static const `openmode ate`
- static const `iosstate badbit`
- static const `fmtflags basefield`
- static const `seekdir beg`
- static const `openmode binary`
- static const `fmtflags boolalpha`
- static const `seekdir cur`
- static const `fmtflags dec`
- static const `seekdir end`
- static const `iosstate eofbit`
- static const `iosstate failbit`
- static const `fmtflags fixed`
- static const `fmtflags floatfield`
- static const `iosstate goodbit`
- static const `fmtflags hex`
- static const `openmode in`

- static const `fmtflags` `internal`
- static const `fmtflags` `left`
- static const `fmtflags` `oct`
- static const `openmode` `out`
- static const `fmtflags` `right`
- static const `fmtflags` `scientific`
- static const `fmtflags` `showbase`
- static const `fmtflags` `showpoint`
- static const `fmtflags` `showpos`
- static const `fmtflags` `skipws`
- static const `openmode` `trunc`
- static const `fmtflags` `unitbuf`
- static const `fmtflags` `uppercase`

### Protected Types

- enum { `_S_local_word_size` }

### Protected Member Functions

- void `_M_cache_locale` (const `locale` &\_\_loc)
- void `_M_call_callbacks` (`event` \_\_ev) throw ()
- void `_M_dispose_callbacks` (void) throw ()
- template<typename \_ValueT >  
    `__istream_type` & `_M_extract` (\_ValueT &\_\_v)
- `_Words` & `_M_grow_words` (int \_\_index, bool \_\_iword)
- void `_M_init` () throw ()
- void `init` (`basic_streambuf`<\_CharT, \_Traits> \*\_\_sb)

### Protected Attributes

- `_Callback_list` \* `_M_callbacks`
- const `__ctype_type` \* `_M_ctype`
- `iostate` `_M_exception`
- `char_type` `_M_fill`
- bool `_M_fill_init`
- `fmtflags` `_M_flags`
- `streamsize` `_M_gcount`
- `locale` `_M_ios_locale`
- `_Words` `_M_local_word` [`_S_local_word_size`]
- const `__num_get_type` \* `_M_num_get`
- const `__num_put_type` \* `_M_num_put`

- [streamsize \\_M\\_precision](#)
- [basic\\_streambuf<\\_CharT, \\_Traits> \\* \\_M\\_streambuf](#)
- [iostate \\_M\\_streambuf\\_state](#)
- [basic\\_ostream<\\_CharT, \\_Traits> \\* \\_M\\_tie](#)
- [streamsize \\_M\\_width](#)
- `_Words * _M_word`
- `int _M_word_size`
- `_Words _M_word_zero`

### Friends

- class `sentry`

#### 5.381.1 Detailed Description

`template<typename _CharT, typename _Traits> class std::basic_istream< _CharT, _Traits >`

Controlling input.

This is the base class for all input streams. It provides text formatting of all builtin types, and communicates with any class derived from [basic\\_streambuf](#) to do the actual input.

Definition at line 57 of file `istream`.

#### 5.381.2 Member Typedef Documentation

**5.381.2.1** `template<typename _CharT, typename _Traits> typedef ctype<_CharT> std::basic_istream< _CharT, _Traits >::__ctype_type`

These are non-standard types.

Reimplemented from `std::basic_ios< _CharT, _Traits >`.

Definition at line 73 of file `istream`.

**5.381.2.2** `template<typename _CharT, typename _Traits> typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits> > std::basic_istream< _CharT, _Traits >::__num_get_type`

These are non-standard types.

Reimplemented from `std::basic_ios< _CharT, _Traits >`.

Definition at line 72 of file `istream`.

**5.381.2.3** `template<typename _CharT, typename _Traits> typedef  
num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>  
> std::basic_ios<_CharT, _Traits>::__num_put_type  
[inherited]`

These are non-standard types.

Reimplemented in `std::basic_ostream<_CharT, _Traits>`, `std::basic_ostream<char, _Traits>`, and `std::basic_ostream<char>`.

Definition at line 86 of file `basic_ios.h`.

**5.381.2.4** `template<typename _CharT, typename _Traits> typedef _CharT  
std::basic_istream<_CharT, _Traits>::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Reimplemented in `std::basic_ifstream<_CharT, _Traits>`, and `std::basic_istreamstream<_CharT, _Traits, _Alloc>`.

Definition at line 61 of file `istream`.

**5.381.2.5** `typedef void(* std::ios_base::event_callback)(event, ios_base &, int)  
[inherited]`

The type of an event callback function.

#### Parameters

***event*** One of the members of the event enum.

***ios\_base*** Reference to the `ios_base` object.

***int*** The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 438 of file `ios_base.h`.

#### 5.381.2.6 `typedef _Ios_Fmtflags std::ios_base::fmtflags` [inherited]

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 257 of file `ios_base.h`.

**5.381.2.7** `template<typename _CharT, typename _Traits> typedef  
_Traits::int_type std::basic_istream<_CharT, _Traits>::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Reimplemented in `std::basic_ifstream<_CharT, _Traits>`, and `std::basic_istreamstream<_CharT, _Traits, _Alloc>`.

Definition at line 62 of file `istream`.

**5.381.2.8** `typedef _Ios_Iostate std::ios_base::iostate [inherited]`

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 332 of file `ios_base.h`.

**5.381.2.9** `template<typename _CharT, typename _Traits> typedef  
_Traits::off_type std::basic_istream<_CharT, _Traits>::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Reimplemented in `std::basic_ifstream<_CharT, _Traits>`, and `std::basic_istreamstream<_CharT, _Traits, _Alloc>`.

Definition at line 64 of file `istream`.

**5.381.2.10 `typedef _Ios_Openmode std::ios_base::openmode` [inherited]**

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 363 of file `ios_base.h`.

**5.381.2.11 `template<typename _CharT, typename _Traits> typedef _Traits::pos_type std::basic_istream<_CharT, _Traits>::pos_type`**

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Reimplemented in `std::basic_ifstream<_CharT, _Traits>`, and `std::basic_istream<_CharT, _Traits, _Alloc>`.

Definition at line 63 of file `istream`.

**5.381.2.12 `typedef _Ios_Seekdir std::ios_base::seekdir` [inherited]**

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.

- end, equivalent to `SEEK_END` in the C standard library.

Definition at line 395 of file `ios_base.h`.

#### **5.381.2.13 template<typename \_CharT, typename \_Traits> typedef \_Traits std::basic\_istream< \_CharT, \_Traits >::traits\_type**

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios< _CharT, _Traits >`.

Reimplemented in `std::basic_ifstream< _CharT, _Traits >`, and `std::basic_istream< _CharT, _Traits, _Alloc >`.

Definition at line 65 of file `istream`.

### **5.381.3 Member Enumeration Documentation**

#### **5.381.3.1 enum std::ios\_base::event [inherited]**

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 421 of file `ios_base.h`.

### **5.381.4 Constructor & Destructor Documentation**

#### **5.381.4.1 template<typename \_CharT, typename \_Traits> std::basic\_istream< \_CharT, \_Traits >::basic\_istream ( \_\_streambuf\_type \* \_\_sb ) [inline, explicit]**

Base constructor.

This ctor is almost never called by the user directly, rather from derived classes' initialization lists, which pass a pointer to their own stream buffer.

Definition at line 93 of file `istream`.



**5.381.4.2** `template<typename _CharT, typename _Traits> virtual  
std::basic_istream<_CharT, _Traits>::~~basic_istream ( )  
[inline, virtual]`

Base destructor.

This does very little apart from providing a virtual base dtor.

Definition at line 103 of file `istream`.

### 5.381.5 Member Function Documentation

**5.381.5.1** `const locale& std::ios_base::_M_getloc ( ) const [inline,  
inherited]`

Locale access.

#### Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 708 of file `ios_base.h`.

Referenced by `std::money_get<_CharT, _InIter>::do_get()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_date()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_time()`, `std::time_get<_CharT, _InIter>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::time_put<_CharT, _OutIter>::do_put()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::time_put<_CharT, _OutIter>::put()`.

**5.381.5.2** `template<typename _CharT, typename _Traits> bool std::basic_ios<  
_CharT, _Traits>::bad ( ) const [inline, inherited]`

Fast error checking.

#### Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 203 of file `basic_ios.h`.

**5.381.5.3** `template<typename _CharT, typename _Traits> void  
std::basic_ios<_CharT, _Traits>::clear ( iostate __state = goodbit )  
[inherited]`

[Re]sets the error state.

#### Parameters

*state* The new state flag(s) to set.

See [std::ios\\_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

Definition at line 42 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::exceptions()`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::rdstate()`.

Referenced by `std::basic_ios<char, _Traits>::exceptions()`, `std::basic_fstream<_CharT, _Traits>::open()`, `std::basic_ofstream<_CharT, _Traits>::open()`, `std::basic_ifstream<_CharT, _Traits>::open()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ios<char, _Traits>::setstate()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

**5.381.5.4** `template<typename _CharT, typename _Traits> basic_ios<  
_CharT, _Traits> & std::basic_ios<_CharT, _Traits>::copyfmt (  
const basic_ios<_CharT, _Traits> & __rhs ) [inherited]`

Copies fields of `__rhs` into this.

#### Parameters

*\_\_rhs* The source values for the copies.

#### Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy [exceptions\(\)](#).

Definition at line 64 of file `basic_ios.tcc`.

References `std::basic_ios< _CharT, _Traits >::exceptions()`, `std::basic_ios< _CharT, _Traits >::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios< _CharT, _Traits >::tie()`, and `std::ios_base::width()`.

#### 5.381.5.5 `template<typename _CharT, typename _Traits> bool std::basic_istream< _CharT, _Traits >::eof( ) const` `[inline, inherited]`

Fast error checking.

##### Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 182 of file `basic_ios.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

#### 5.381.5.6 `template<typename _CharT, typename _Traits> void std::basic_istream< _CharT, _Traits >::exceptions( iostate __except )` `[inline, inherited]`

Throwing exceptions on errors.

##### Parameters

*except* The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>
```

```
int main()
{
 std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

 std::ifstream f ("/etc/motd");

 std::cerr << "Setting badbit\n";
 f.setstate (std::ios_base::badbit);

 std::cerr << "Setting exception mask\n";
 f.exceptions (std::ios_base::badbit);
}
```

Definition at line 249 of file `basic_ios.h`.

**5.381.5.7** `template<typename _CharT, typename _Traits> iostate  
std::basic_ios<_CharT, _Traits>::exceptions ( ) const [inline,  
inherited]`

Throwing exceptions on errors.

#### Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of [exceptions\(iostate\)](#) for the meaning of the return value.

Definition at line 214 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::clear()`, and `std::basic_ios<_CharT, _Traits>::copyfmt()`.

**5.381.5.8** `template<typename _CharT, typename _Traits> bool std::basic_ios<  
_CharT, _Traits>::fail ( ) const [inline, inherited]`

Fast error checking.

#### Returns

True if either the badbit or the failbit is set.

Checking the badbit in [fail\(\)](#) is historical practice. Note that other iostate flags may also be set.

Definition at line 193 of file `basic_ios.h`.

Referenced by `std::basic_ios<char, _Traits>::operator void*()`, `std::basic_ios<char, _Traits>::operator!()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_ostream<_CharT, _Traits>::tellp()`, and `std::regex_traits<_Ch_type>::value()`.

**5.381.5.9** `template<typename _CharT, typename _Traits> char_type  
std::basic_ios<_CharT, _Traits>::fill ( char_type __ch )  
[inline, inherited]`

Sets a new *empty* character.

#### Parameters

*ch* The new character.

#### Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 382 of file `basic_ios.h`.

**5.381.5.10** `template<typename _CharT, typename _Traits> char_type  
std::basic_ios<_CharT, _Traits>::fill ( ) const [inline,  
inherited]`

Retrieves the *empty* character.

#### Returns

The current fill character.

It defaults to a space ( ' ') in the current locale.

Definition at line 362 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, and `std::basic_ios<char, _Traits>::fill()`.

**5.381.5.11 `fmtflags std::ios_base::flags ( ) const [inline, inherited]`**

Access to format flags.

**Returns**

The format control flags for both input and output.

Definition at line 553 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::operator<<()`, `std::operator>>()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

**5.381.5.12 `fmtflags std::ios_base::flags ( fmtflags __fmtfl ) [inline, inherited]`**

Setting new format flags all at once.

**Parameters**

*fmtfl* The new flags to set.

**Returns**

The previous format control flags.

This function overwrites all the format flags with *fmtfl*.

Definition at line 564 of file `ios_base.h`.

**5.381.5.13 `template<typename _CharT, typename _Traits> streamsize std::basic_istream<_CharT, _Traits>::gcount ( ) const [inline]`**

Character counting.

**Returns**

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 251 of file `istream`.

**5.381.5.14** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::get ( void )`

Simple extraction.

#### Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 238 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.381.5.15** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get ( char_type & __c )`

Simple extraction.

#### Parameters

*c* The character in which to store data.

#### Returns

`*this`

Tries to extract a character and store it in *c*. If none are available, sets failbit and returns `traits::eof()`.

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 274 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.381.5.16** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get ( char_type * __s, streamsize __n, char_type __delim )`

Simple multiple-character extraction.

#### Parameters

- s* Pointer to an array.
- n* Maximum number of characters to store in *s*.
- delim* A "stop" character.

#### Returns

\*this

Characters are extracted and stored into *s* until one of the following happens:

- *n*-1 characters are stored
- the input sequence reaches EOF
- the next character equals *delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 311 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

**5.381.5.17** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::get ( char_type * __s, streamsize __n ) [inline]`

Simple multiple-character extraction.



**Parameters**

- s* Pointer to an array.  
*n* Maximum number of characters to store in *s*.

**Returns**

\*this

Returns `get(s,n,widen('\n'))`.

Definition at line 335 of file `istream`.

**5.381.5.18** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get ( __streambuf_type & __sb, char_type __delim )`

Extraction into another streambuf.

**Parameters**

- sb* A streambuf in which to store data.  
*delim* A "stop" character.

**Returns**

\*this

Characters are extracted and inserted into *sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 358 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, `std::basic_streambuf<_CharT, _Traits>::snextc()`, and `std::basic_streambuf<_CharT, _Traits>::sputc()`.

**5.381.5.19** `template<typename _CharT, typename _Traits> __istream_type&  
std::basic_istream<_CharT, _Traits>::get ( __streambuf_type &  
__sb ) [inline]`

Extraction into another streambuf.

#### Parameters

*sb* A streambuf in which to store data.

#### Returns

\*this

Returns `get(sb, widen('\n'))`.

Definition at line 368 of file `istream`.

**5.381.5.20** `template<typename _CharT, typename _Traits> basic_istream<  
_CharT, _Traits> & std::basic_istream<_CharT, _Traits  
>::getline ( char_type * __s, streamsize __n, char_type __delim )`

String extraction.

#### Parameters

*s* A character array in which to store the data.

*n* Maximum number of characters to extract.

*delim* A "stop" character.

#### Returns

\*this

Extracts and stores characters into *s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case `eofbit` is set in the stream error state
2. the next character equals *delim*, in which case the character is extracted (and therefore counted in `gcount()`) but not stored

3.  $n-1$  characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 402 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sbumpc()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::sngetc()`.

**5.381.5.21** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::getline ( char_type * __s, streamsize __n ) [inline]`

String extraction.

#### Parameters

- s* A character array in which to store the data.
- n* Maximum number of characters to extract.

#### Returns

*\*this*

Returns `getline(s,n,widen('\n'))`.

Definition at line 408 of file `istream`.

Referenced by `std::basic_istream<char>::getline()`.

**5.381.5.22** `locale std::ios_base::getloc ( ) const [inline, inherited]`

Locale access.

#### Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 697 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::money_put<_CharT, _OutputIter>::do_put()`, `std::basic_ios<_CharT, _Traits>::imbue()`, `std::operator>>()`, and `std::ws()`.

**5.381.5.23** `template<typename _CharT, typename _Traits> bool  
std::basic_ios<_CharT, _Traits>::good( ) const [inline,  
inherited]`

Fast error checking.

#### Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 172 of file `basic_ios.h`.

Referenced by `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

**5.381.5.24** `template<typename _CharT, typename _Traits> basic_istream<  
_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore  
( void )`

Discarding characters.

#### Parameters

*n* Number of characters to discard.

*delim* A "stop" character.

#### Returns

`*this`

Extracts characters and throws them away until one of the following happens:

- if *n* != `std::numeric_limits<int>::max()`, *n* characters are extracted

- the input sequence reaches end-of-file
- the next character equals *delim* (in this case, the character is extracted); note that this condition will never occur if *delim* equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 462 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_streambuf< _CharT, _Traits >::sbumpc()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.381.5.25** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore ( streamsize __n )`

Simple extraction.

#### Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 495 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

**5.381.5.26** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore ( streamsize __n, int_type __delim )`

Simple extraction.

#### Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 557 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_streambuf< _CharT, _Traits >::sbumpc()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

**5.381.5.27** `template<typename _CharT, typename _Traits > locale  
std::basic_ios< _CharT, _Traits >::imbue ( const locale & __loc )  
[inherited]`

Moves to a new locale.

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Reimplemented from `std::ios_base`.

Definition at line 115 of file `basic_ios.tcc`.

References `std::ios_base::getloc()`, and `std::basic_ios< _CharT, _Traits >::rdbuf()`.

Referenced by `std::operator<<()`.

**5.381.5.28** `template<typename _CharT, typename _Traits> void  
std::basic_ios< _CharT, _Traits >::init ( basic_streambuf<  
_CharT, _Traits > * __sb ) [protected, inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 127 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

Referenced by `std::basic_fstream<_CharT, _Traits>::basic_fstream()`, `std::basic_ifstream<_CharT, _Traits>::basic_ifstream()`, `std::basic_ios<char, _Traits>::basic_ios()`, `std::basic_istream<char>::basic_istream()`, `std::basic_istreamstream<_CharT, _Traits, _Alloc>::basic_istreamstream()`, `std::basic_ofstream<_CharT, _Traits>::basic_ofstream()`, `std::basic_ostream<char>::basic_ostream()`, `std::basic_ostreamstream<_CharT, _Traits, _Alloc>::basic_ostreamstream()`, and `std::basic_stringstream<_CharT, _Traits, _Alloc>::basic_stringstream()`.

#### 5.381.5.29 `long& std::ios_base::iword ( int __ix ) [inline, inherited]`

Access to integer array.

##### Parameters

`__ix` Index into the array.

##### Returns

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 743 of file `ios_base.h`.

#### 5.381.5.30 `template<typename _CharT, typename _Traits> char std::basic_ios<_CharT, _Traits>::narrow ( char_type __c, char __dfault ) const [inline, inherited]`

Squeezes characters.

##### Parameters

`c` The character to narrow.

`dfault` The character to narrow.

##### Returns

The narrowed character.

## 5.381 `std::basic_istream<_CharT, _Traits>` Class Template Reference 1917

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, default)
```

Additional notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 422 of file `basic_ios.h`.

**5.381.5.31** `template<typename _CharT, typename _Traits> std::basic_ios<_CharT, _Traits>::operator void * ( ) const` [`inline`, `inherited`]

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 113 of file `basic_ios.h`.

**5.381.5.32** `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::operator! ( ) const` [`inline`, `inherited`]

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 117 of file `basic_ios.h`.

**5.381.5.33** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( unsigned int & __n )` [`inline`]

Basic arithmetic extractors.

### Parameters

`A` variable of builtin type.

### Returns

`*this` if successful



These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 183 of file `istream`.

**5.381.5.34** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::operator>> ( __streambuf_type * __sb )`

Extracting into another streambuf.

#### Parameters

*sb* A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the *sb* streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 206 of file `istream.tcc`.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.381.5.35** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::operator>> ( short & __n )`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 116 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get<_CharT, _InIter>::get()`, `std::ios_base::goodbit`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.381.5.36** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( __istream_type &(*)(__istream_type &) __pf ) [inline]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 122 of file `istream`.

**5.381.5.37** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( long & __n ) [inline]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 187 of file `istream`.

**5.381.5.38** `template<typename _CharT, typename _Traits> __istream_type&  
std::basic_istream<_CharT, _Traits>::operator>> ( long long &  
__n ) [inline]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 196 of file `istream`.

**5.381.5.39** `template<typename _CharT, typename _Traits> __istream_type&  
std::basic_istream<_CharT, _Traits>::operator>> ( __ios_type  
&(*)(__ios_type &) __pf ) [inline]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iomanip` header.

Definition at line 126 of file `istream`.

**5.381.5.40** `template<typename _CharT, typename _Traits> __istream_type&  
std::basic_istream<_CharT, _Traits>::operator>> ( bool & __n  
) [inline]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 169 of file `istream`.

**5.381.5.41** `template<typename _CharT, typename _Traits> __istream_type&  
std::basic_istream<_CharT, _Traits>::operator>> ( unsigned  
long & __n ) [inline]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

\*`this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 191 of file `istream`.

**5.381.5.42** `template<typename _CharT, typename _Traits> __istream_type&  
std::basic_istream<_CharT, _Traits>::operator>> ( ios_base  
&(*)(ios_base &) __pf ) [inline]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iomanip` header.

Definition at line 133 of file `istream`.

**5.381.5.43** `template<typename _CharT, typename _Traits> __istream_type&  
std::basic_istream<_CharT, _Traits>::operator>> ( unsigned  
long long & __n ) [inline]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 200 of file `istream`.

**5.381.5.44** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( unsigned short & __n ) [inline]`

Basic arithmetic extractors.

**Parameters**

`A` variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 176 of file `istream`.

**5.381.5.45** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( void *& __p ) [inline]`

Basic arithmetic extractors.

**Parameters**

`A` variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 217 of file `istream`.

**5.381.5.46** `template<typename _CharT, typename _Traits> __istream_type&  
std::basic_istream<_CharT, _Traits>::operator>> ( float & __f  
) [inline]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 205 of file `istream`.

**5.381.5.47** `template<typename _CharT, typename _Traits> __istream_type&  
std::basic_istream<_CharT, _Traits>::operator>> ( double &  
__f ) [inline]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 209 of file `istream`.

**5.381.5.48** `template<typename _CharT, typename _Traits> basic_istream<  
_CharT, _Traits> & std::basic_istream<_CharT, _Traits  
>::operator>> ( int & __n )`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 161 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get<_CharT, _InIter>::get()`, `std::ios_base::goodbit`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.381.5.49** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( long double & __f ) [inline]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 213 of file `istream`.

**5.381.5.50** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits>::peek ( void )`

Looking ahead in the stream.

**Returns**

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 622 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

#### 5.381.5.51 `streamsize std::ios_base::precision ( streamsize __prec )` [inline, inherited]

Changing flags.

##### Parameters

*prec* The new precision value.

##### Returns

The previous value of `precision()`.

Definition at line 632 of file `ios_base.h`.

#### 5.381.5.52 `streamsize std::ios_base::precision ( ) const` [inline, inherited]

Flags access.

##### Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 623 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, and `std::operator<<()`.

#### 5.381.5.53 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::putback ( char_type __c )`



Unextracting a single character.

#### Parameters

*c* The character to push back into the input stream.

#### Returns

`*this`

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

#### Note

Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 713 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::clear()`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::rdstate()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sputbackc()`.

Referenced by `std::operator>>()`.

#### 5.381.5.54 `void*& std::ios_base::pword ( int __ix ) [inline, inherited]`

Access to void pointer array.

#### Parameters

*\_\_ix* Index into the array.

#### Returns

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 764 of file `ios_base.h`.

```
5.381.5.55 template<typename _CharT, typename _Traits>
 basic_streambuf<_CharT, _Traits>* std::basic_ios<_CharT,
 _Traits>::rdbuf() const [inline, inherited]
```

Accessing the underlying buffer.

#### Returns

The current stream buffer.

This does not change the state of the stream.

Reimplemented in [std::basic\\_ifstream<\\_CharT, \\_Traits>](#), [std::basic\\_ofstream<\\_CharT, \\_Traits>](#), [std::basic\\_fstream<\\_CharT, \\_Traits>](#), [std::basic\\_istream<\\_CharT, \\_Traits, \\_Alloc>](#), [std::basic\\_ostringstream<\\_CharT, \\_Traits, \\_Alloc>](#), and [std::basic\\_stringstream<\\_CharT, \\_Traits, \\_Alloc>](#).

Definition at line 313 of file `basic_ios.h`.

Referenced by `std::basic_ostream<char>::M_write()`, `std::basic_ios<_CharT, _Traits>::clear()`, `std::basic_ostream<_CharT, _Traits>::flush()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_ios<_CharT, _Traits>::imbue()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_ostream<_CharT, _Traits>::put()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::readsome()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, `std::basic_istream<_CharT, _Traits>::sync()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_ostream<_CharT, _Traits>::tellp()`, `std::basic_istream<_CharT, _Traits>::unset()`, and `std::ws()`.

```
5.381.5.56 template<typename _CharT, typename _Traits> basic_streambuf<
 _CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rdbuf (
 basic_streambuf<_CharT, _Traits> * __sb) [inherited]
```

Changing the underlying buffer.

#### Parameters

*sb* The new stream buffer.

#### Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream foo; // or some other derived type
std::streambuf* p =;

foo.ios::rdbuf(p); // ios == basic_ios<char>
```

Definition at line 54 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::clear()`.

**5.381.5.57** `template<typename _CharT, typename _Traits> iostate  
std::basic_ios<_CharT, _Traits>::rdstate( ) const [inline,  
inherited]`

Returns the error state of the stream buffer.

#### Returns

A bit pattern (well, isn't everything?)

See [std::ios\\_base::iostate](#) for the possible bit values. Most users will call one of the interpreting wrappers, e.g., [good\(\)](#).

Definition at line 129 of file `basic_ios.h`.

Referenced by `std::basic_ios<char, _Traits>::bad()`, `std::basic_ios<_CharT, _Traits>::clear()`, `std::basic_ios<char, _Traits>::eof()`, `std::basic_ios<char, _Traits>::fail()`, `std::basic_ios<char, _Traits>::good()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ios<char, _Traits>::setstate()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

**5.381.5.58** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::read ( char_type * __s, streamsize __n )`

Extraction without delimiters.

#### Parameters

- s* A character array.
- n* Maximum number of characters to store.

#### Returns

\*this

If the stream state is `good()`, extracts characters and stores them into *s* until one of the following happens:

- *n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 652 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.381.5.59** `template<typename _CharT, typename _Traits> streamsize std::basic_istream<_CharT, _Traits>::readsome ( char_type * __s, streamsize __n )`

Extraction until the buffer is exhausted, but no more.

#### Parameters

- s* A character array.
- n* Maximum number of characters to store.

**Returns**

The number of characters extracted.

Extracts characters and stores them into *s* depending on the number of characters remaining in the streambuf's buffer, `rddbuf() -> in_avail()`, called *A* here:

- if *A* == -1, sets eofbit and extracts no characters
- if *A* == 0, extracts no characters
- if *A* > 0, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 681 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::min()`, `std::basic_ios<_CharT, _Traits>::rddbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.381.5.60** `void std::ios_base::register_callback ( event_callback __fn, int __index ) [inherited]`

Add the callback `__fn` with parameter `__index`.

**Parameters**

`__fn` The function to add.

`__index` The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**5.381.5.61** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg ( pos_type __pos )`

Changing the current read position.

**Parameters**

*pos* A file position object.

**Returns**

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 847 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::clear()`, `std::ios_base::eofbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::rdstate()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.381.5.62** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg( off_type __off, ios_base::seekdir __dir )`

Changing the current read position.

**Parameters**

*off* A file offset object.

*dir* The direction in which to seek.

**Returns**

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 886 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::clear()`, `std::ios_base::eofbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`,

std::ios\_base::goodbit, std::ios\_base::in, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), std::basic\_ios< \_CharT, \_Traits >::rdstate(), and std::basic\_ios< \_CharT, \_Traits >::setstate().

#### 5.381.5.63 fmtflags std::ios\_base::setf ( fmtflags \_\_*fmtfl* ) [inline, inherited]

Setting new format flags.

##### Parameters

*fmtfl* Additional flags to set.

##### Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 580 of file ios\_base.h.

Referenced by std::dec(), std::fixed(), std::hex(), std::left(), std::oct(), std::right(), std::scientific(), std::showbase(), std::showpoint(), std::showpos(), std::skipws(), std::unitbuf(), and std::uppercase().

#### 5.381.5.64 fmtflags std::ios\_base::setf ( fmtflags \_\_*fmtfl*, fmtflags \_\_*mask* ) [inline, inherited]

Setting new format flags.

##### Parameters

*fmtfl* Additional flags to set.

*mask* The flags mask for *fmtfl*.

##### Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is [ios\\_base::adjustfield](#).

Definition at line 597 of file ios\_base.h.

**5.381.5.65** `template<typename _CharT, typename _Traits> void  
std::basic_ios<_CharT, _Traits>::setstate ( iostate __state )  
[inline, inherited]`

Sets additional flags in the error state.

#### Parameters

*state* The additional state flag(s) to set.

See [std::ios\\_base::iostate](#) for the possible bit values.

Definition at line 149 of file `basic_ios.h`.

Referenced by `std::basic_ostream<char>::_M_write()`, `std::basic_fstream<_CharT, _Traits>::close()`, `std::basic_ofstream<_CharT, _Traits>::close()`, `std::basic_ifstream<_CharT, _Traits>::close()`, `std::basic_ostream<_CharT, _Traits>::flush()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_fstream<_CharT, _Traits>::open()`, `std::basic_ofstream<_CharT, _Traits>::open()`, `std::basic_ifstream<_CharT, _Traits>::open()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_ostream<_CharT, _Traits>::put()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::readsome()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, `std::basic_istream<_CharT, _Traits>::sync()`, `std::basic_istream<_CharT, _Traits>::unget()`, and `std::ws()`.

**5.381.5.66** `template<typename _CharT, typename _Traits> int  
std::basic_istream<_CharT, _Traits>::sync ( void )`

Synchronizing the stream buffer.

#### Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf() ->pubsync()`, and if that returns -1, sets `badbit` and returns -1.



Otherwise, returns 0.

#### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 783 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_streambuf<_CharT, _Traits>::pubsync()`, `std::basic_istream<_CharT, _Traits>::rdbuf()`, and `std::basic_istream<_CharT, _Traits>::setstate()`.

**5.381.5.67** `static bool std::ios_base::sync_with_stdio ( bool __sync = true )`  
[static, inherited]

Interaction with the standard C I/O objects.

#### Parameters

*sync* Whether to synchronize or not.

#### Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

**5.381.5.68** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::pos_type std::basic_istream<_CharT, _Traits>::tellg ( void )`

Getting the current read position.

#### Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, in)`.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 819 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::in`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

**5.381.5.69** `template<typename _CharT, typename _Traits>  
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie( ) const [inline, inherited]`

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 287 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

**5.381.5.70** `template<typename _CharT, typename _Traits>  
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie( basic_ostream<_CharT, _Traits> * __tiestr ) [inline, inherited]`

Ties this stream to an output stream.

**Parameters**

*tiestr* The output stream.

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see [tie\(\)](#) for more.

Definition at line 299 of file basic\_ios.h.

**5.381.5.71** `template<typename _CharT, typename _Traits > basic_istream<  
_CharT, _Traits > & std::basic_istream< _CharT, _Traits >::unget  
( void )`

Unextracting the previous character.

#### Returns

\*this

If [rdbuf\(\)](#) is not null, calls [rdbuf\(\)->sungetc\(c\)](#).

If [rdbuf\(\)](#) is null or if [sungetc\(\)](#) fails, sets badbit in the error state.

#### Note

Since no characters are extracted, the next call to [gcount\(\)](#) will return 0, as required by DR 60.

Definition at line 748 of file istream.tcc.

References [std::basic\\_istream< \\_CharT, \\_Traits >::M\\_gcount](#), [std::ios\\_base::badbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::clear\(\)](#), [std::basic\\_ios< \\_CharT, \\_Traits >::eof\(\)](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdstate\(\)](#), [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#), and [std::basic\\_streambuf< \\_CharT, \\_Traits >::sungetc\(\)](#).

**5.381.5.72** `void std::ios_base::unsetf( fmtflags __mask ) [inline,  
inherited]`

Clearing format flags.

#### Parameters

*mask* The flags to unset.

This function clears *mask* in the format flags.

Definition at line 612 of file ios\_base.h.

Referenced by [std::noboolalpha\(\)](#), [std::noshowbase\(\)](#), [std::noshowpoint\(\)](#), [std::noshowpos\(\)](#), [std::noskipws\(\)](#), [std::nounitbuf\(\)](#), and [std::nouppercase\(\)](#).

**5.381.5.73** `template<typename _CharT, typename _Traits> char_type  
std::basic_ios<_CharT, _Traits>::widen ( char __c ) const  
[inline, inherited]`

Widens characters.

#### Parameters

*c* The character to widen.

#### Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 441 of file `basic_ios.h`.

Referenced by `std::endl()`, `std::basic_ios<char, _Traits>::fill()`, `std::basic_istream<char>::get()`, `std::basic_istream<char>::getline()`, `std::getline()`, and `std::operator>>()`.

**5.381.5.74** `streamsize std::ios_base::width ( streamsize __wide ) [inline,  
inherited]`

Changing flags.

#### Parameters

*wide* The new width value.

#### Returns

The previous value of [width\(\)](#).

Definition at line 655 of file `ios_base.h`.

**5.381.5.75** streamsize std::ios\_base::width ( ) const [inline, inherited]

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 646 of file ios\_base.h.

Referenced by std::basic\_ios<\_CharT, \_Traits>::copyfmt(), std::num\_put<\_CharT, \_OutIter>::do\_put(), and std::operator>>().

**5.381.5.76** static int std::ios\_base::xalloc ( ) throw () [static, inherited]

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the iword and pword functions. The expectation is that an application calls xalloc in order to obtain an index in the iword and pword arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. xalloc is guaranteed to return an index that is safe to use in the iword and pword arrays.

**5.381.6 Member Data Documentation****5.381.6.1** template<typename \_CharT, typename \_Traits> streamsize std::basic\_istream<\_CharT, \_Traits>::\_M\_gcount [protected]

The number of characters extracted in the previous unformatted function; see [gcount\(\)](#).

Definition at line 81 of file istream.

Referenced by `std::basic_istream< char >::gcount()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::basic_istream< char >::~~basic_istream()`.

#### **5.381.6.2 const fmtflags std::ios\_base::adjustfield [static, inherited]**

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 312 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

#### **5.381.6.3 const openmode std::ios\_base::app [static, inherited]**

Seek to end before each write.

Definition at line 366 of file `ios_base.h`.

#### **5.381.6.4 const openmode std::ios\_base::ate [static, inherited]**

Open and seek to end immediately after opening.

Definition at line 369 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

#### **5.381.6.5 const iostate std::ios\_base::badbit [static, inherited]**

Indicates a loss of integrity in an input or output sequence (such /// as an irrecoverable read error from a file).

Definition at line 336 of file `ios_base.h`.

Referenced by `std::basic_ostream< char >::_M_write()`, `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::basic_ios< char,`

`_Traits >::fail()`, `std::basic_ostream<_CharT, _Traits >::flush()`, `std::basic_istream<_CharT, _Traits >::get()`, `std::basic_istream<_CharT, _Traits >::getline()`, `std::basic_istream<_CharT, _Traits >::ignore()`, `std::basic_ios<_CharT, _Traits >::init()`, `std::basic_ostream<_CharT, _Traits >::operator<<()`, `std::operator<<()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits >::operator>>()`, `std::basic_istream<_CharT, _Traits >::peek()`, `std::basic_ostream<_CharT, _Traits >::put()`, `std::basic_istream<_CharT, _Traits >::putback()`, `std::basic_istream<_CharT, _Traits >::read()`, `std::basic_istream<_CharT, _Traits >::readsome()`, `std::basic_istream<_CharT, _Traits >::seekg()`, `std::basic_ostream<_CharT, _Traits >::seekp()`, `std::basic_istream<_CharT, _Traits >::sync()`, `std::basic_istream<_CharT, _Traits >::tellg()`, `std::basic_ostream<_CharT, _Traits >::tellp()`, `std::basic_istream<_CharT, _Traits >::unget()`, `std::basic_ostream<_CharT, _Traits >::write()`, and `std::basic_ostream<_CharT, _Traits >::sentry::~sentry()`.

#### 5.381.6.6 `const fmtflags std::ios_base::basefield` [`static`, `inherited`]

A mask of `dec|oct|hex`. Useful for the 2-arg form of `setf`.

Definition at line 315 of file `ios_base.h`.

Referenced by `std::dec()`, `std::num_get<_CharT, _InIter >::do_get()`, `std::hex()`, `std::oct()`, and `std::basic_ostream<_CharT, _Traits >::operator<<()`.

#### 5.381.6.7 `const seekdir std::ios_base::beg` [`static`, `inherited`]

Request a seek relative to the beginning of the stream.

Definition at line 398 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits >::seekpos()`.

#### 5.381.6.8 `const openmode std::ios_base::binary` [`static`, `inherited`]

Perform input and output in binary mode (as opposed to text mode). `/// This is probably not what you think it is; see http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html.`

Definition at line 374 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits >::showmanyc()`.

**5.381.6.9 const fmtflags std::ios\_base::boolalpha [static, inherited]**

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 260 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::noboolalpha()`.

**5.381.6.10 const seekdir std::ios\_base::cur [static, inherited]**

Request a seek relative to the current position within the sequence.

Definition at line 401 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::pbackfail()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_filebuf<_CharT, _Traits>::seekoff()`, `std::basic_istream<_CharT, _Traits>::tellg()`, and `std::basic_ostream<_CharT, _Traits>::tellp()`.

**5.381.6.11 const fmtflags std::ios\_base::dec [static, inherited]**

Converts integer input or generates integer output in decimal base.

Definition at line 263 of file `ios_base.h`.

Referenced by `std::dec()`.

**5.381.6.12 const seekdir std::ios\_base::end [static, inherited]**

Request a seek relative to the current end of the sequence.

Definition at line 404 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::open()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`.

**5.381.6.13 const iostate std::ios\_base::eofbit [static, inherited]**



Indicates that an input operation reached the end of an input sequence.

Definition at line 339 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_date(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::eof(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::unget(), and std::ws().

#### 5.381.6.14 const iostate std::ios\_base::failbit [static, inherited]

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 344 of file ios\_base.h.

Referenced by std::basic\_fstream< \_CharT, \_Traits >::close(), std::basic\_ofstream< \_CharT, \_Traits >::close(), std::basic\_ifstream< \_CharT, \_Traits >::close(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::fail(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_fstream< \_CharT, \_Traits >::open(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry(), and std::basic\_istream< \_CharT, \_Traits >::sentry::sentry().

#### 5.381.6.15 const fmtflags std::ios\_base::fixed [static, inherited]

Generate floating-point output in fixed-point notation.

Definition at line 266 of file ios\_base.h.

Referenced by `std::fixed()`.

#### **5.381.6.16 const fmtflags std::ios\_base::floatfield [static, inherited]**

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 318 of file `ios_base.h`.

Referenced by `std::fixed()`, and `std::scientific()`.

#### **5.381.6.17 const iostate std::ios\_base::goodbit [static, inherited]**

Indicates all is well.

Definition at line 347 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::basic_ostream<_CharT, _Traits>::flush()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_ios<_CharT, _Traits>::init()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_ostream<_CharT, _Traits>::put()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::readsome()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, `std::basic_istream<_CharT, _Traits>::sync()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

#### **5.381.6.18 const fmtflags std::ios\_base::hex [static, inherited]**

Converts integer input or generates integer output in hexadecimal base.

Definition at line 269 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, `std::hex()`, and `std::basic_ostream<_CharT, _Traits>::operator<<()`.

**5.381.6.19 const openmode std::ios\_base::in [static, inherited]**

Open for input. Default for `ifstream` and `fstream`.

Definition at line 377 of file `ios_base.h`.

Referenced by `std::basic_filebuf<char_type, traits_type>::_M_set_buffer()`, `std::basic_ifstream<_CharT, _Traits>::open()`, `std::basic_filebuf<_CharT, _Traits>::_pbackfail()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::showmanyc()`, `std::basic_filebuf<_CharT, _Traits>::showmanyc()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::underflow()`, `std::basic_filebuf<_CharT, _Traits>::underflow()`, and `std::basic_filebuf<_CharT, _Traits>::xsgetn()`.

**5.381.6.20 const fmtflags std::ios\_base::internal [static, inherited]**

Adds fill characters at a designated internal point in certain `///` generated output, or identical to `right` if no such point is `///` designated.

Definition at line 274 of file `ios_base.h`.

Referenced by `std::internal()`.

**5.381.6.21 const fmtflags std::ios\_base::left [static, inherited]**

Adds fill characters on the right (final positions) of certain `///` generated output. (I.e., the thing you print is flush left.).

Definition at line 278 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`, and `std::left()`.

**5.381.6.22 const fmtflags std::ios\_base::oct [static, inherited]**

Converts integer input or generates integer output in octal base.

Definition at line 281 of file `ios_base.h`.

Referenced by `std::oct()`, and `std::basic_ostream<_CharT, _Traits>::operator<<()`.

**5.381.6.23 `const openmode std::ios_base::out` [`static`, `inherited`]**

Open for output. Default for `ofstream` and `fstream`.

Definition at line 380 of file `ios_base.h`.

Referenced by `std::basic_filebuf<char_type, traits_type>::_M_set_buffer()`, `std::basic_ofstream<_CharT, _Traits>::open()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::pbackfail()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, `std::basic_ostream<_CharT, _Traits>::tellp()`, and `std::basic_filebuf<_CharT, _Traits>::xsputn()`.

**5.381.6.24 `const fmtflags std::ios_base::right` [`static`, `inherited`]**

Adds fill characters on the left (initial positions) of certain `///` generated output. (I.e., the thing you print is flush right.).

Definition at line 285 of file `ios_base.h`.

Referenced by `std::right()`.

**5.381.6.25 `const fmtflags std::ios_base::scientific` [`static`, `inherited`]**

Generates floating-point output in scientific notation.

Definition at line 288 of file `ios_base.h`.

Referenced by `std::scientific()`.

**5.381.6.26 `const fmtflags std::ios_base::showbase` [`static`, `inherited`]**

Generates a prefix indicating the numeric base of generated integer `///` output.

Definition at line 292 of file `ios_base.h`.

Referenced by `std::noshowbase()`, and `std::showbase()`.

**5.381.6.27** `const fmtflags std::ios_base::showpoint` [`static`, `inherited`]

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 296 of file `ios_base.h`.

Referenced by `std::noshowpoint()`, and `std::showpoint()`.

**5.381.6.28** `const fmtflags std::ios_base::showpos` [`static`, `inherited`]

Generates a + sign in non-negative generated numeric output.

Definition at line 299 of file `ios_base.h`.

Referenced by `std::noshowpos()`, and `std::showpos()`.

**5.381.6.29** `const fmtflags std::ios_base::skipws` [`static`, `inherited`]

Skips leading white space before certain input operations.

Definition at line 302 of file `ios_base.h`.

Referenced by `std::noskipws()`, `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, and `std::skipws()`.

**5.381.6.30** `const openmode std::ios_base::trunc` [`static`, `inherited`]

Open for input. Default for `ofstream`.

Definition at line 383 of file `ios_base.h`.

**5.381.6.31** `const fmtflags std::ios_base::unitbuf` [`static`, `inherited`]

Flushes output after each output operation.

Definition at line 305 of file `ios_base.h`.

Referenced by `std::nounitbuf()`, `std::unitbuf()`, and `std::basic_ostream<_CharT, _Traits>::sentry::~sentry()`.

**5.381.6.32 `const fmtflags std::ios_base::uppercase` [static, inherited]**

Replaces certain lowercase letters with their uppercase equivalents /// in generated output.

Definition at line 309 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, `std::nouppercase()`, and `std::uppercase()`.

The documentation for this class was generated from the following files:

- [istream](#)
- [istream.tcc](#)

**5.382 `std::basic_istream< _CharT, _Traits >::sentry` Class Reference**

Performs setup work for input streams.

**Public Types**

- typedef `__istream_type::__ctype_type` `__ctype_type`
- typedef `_Traits::int_type` `__int_type`
- typedef `basic_istream< _CharT, _Traits >` `__istream_type`
- typedef `basic_streambuf< _CharT, _Traits >` `__streambuf_type`
- typedef `_Traits` `traits_type`

**Public Member Functions**

- `sentry(basic_istream< _CharT, _Traits > &__is, bool __noskipws=false)`
- `operator bool() const`

**5.382.1 Detailed Description**

**template<typename \_CharT, typename \_Traits> class `std::basic_istream< _CharT, _Traits >::sentry`**

Performs setup work for input streams. Objects of this class are created before all of the standard extractors are run. It is responsible for *exception-safe prefix and suffix operations*, although only prefix actions are currently required by the standard.

Definition at line 635 of file `istream`.

### 5.382.2 Member Typedef Documentation

#### 5.382.2.1 template<typename \_CharT, typename \_Traits> typedef \_Traits std::basic\_istream< \_CharT, \_Traits >::sentry::traits\_type

Easy access to dependant types.

Definition at line 642 of file istream.

### 5.382.3 Constructor & Destructor Documentation

#### 5.382.3.1 template<typename \_CharT, typename \_Traits> std::basic\_istream< \_CharT, \_Traits >::sentry::sentry ( basic\_istream< \_CharT, \_Traits > & \_\_is, bool \_\_noskipws = false ) [explicit]

The constructor performs all the work.

#### Parameters

*is* The input stream to guard.

*noskipws* Whether to consume whitespace or not.

If the stream state is good (*is.good()* is true), then the following actions are performed, otherwise the sentry state is false (*not okay*) and failbit is set in the stream state.

The sentry's preparatory actions are:

1. if the stream is tied to an output stream, `is.tie()->flush()` is called to synchronize the output sequence
2. if *noskipws* is false, and `ios_base::skipws` is set in `is.flags()`, the sentry extracts and discards whitespace characters from the stream. The currently imbued locale is used to determine whether each character is whitespace.

If the stream state is still good, then the sentry state becomes true (*okay*).

Definition at line 49 of file istream.tcc.

References `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::flags()`, `std::basic_ios< _CharT, _Traits >::good()`, `std::ios_base::goodbit`, `std::__ctype_abstract_base< _CharT >::is()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, `std::ios_base::skipws`, `std::basic_streambuf< _CharT, _Traits >::nextc()`, and `std::basic_ios< _CharT, _Traits >::tie()`.

#### 5.382.4 Member Function Documentation

**5.382.4.1** `template<typename _CharT, typename _Traits> std::basic_istream< _CharT, _Traits >::sentry::operator bool ( ) const` [`inline`, `explicit`]

Quick status checking.

##### Returns

The sentry state.

For ease of use, sentries may be converted to booleans. The return value is that of the sentry state (`true == okay`).

Definition at line 683 of file `istream`.

The documentation for this class was generated from the following files:

- [istream](#)
- [istream.tcc](#)

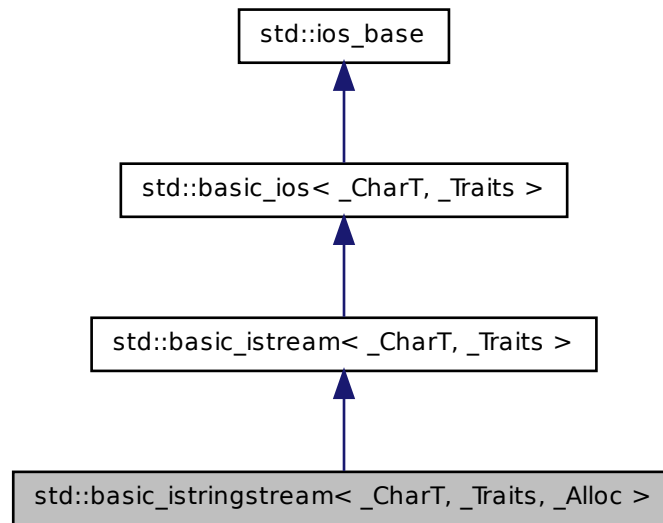
### 5.383 `std::basic_istream< _CharT, _Traits, _Alloc >` Class Template Reference

Controlling input for `std::string`.

This class supports reading from objects of type [std::basic\\_string](#), using the inherited functions from [std::basic\\_istream](#). To control the associated sequence, an instance of [std::basic\\_stringbuf](#) is used, which this page refers to as `sb`.



Inheritance diagram for `std::basic_istream< _CharT, _Traits, _Alloc >`:



### Public Types

- typedef `ctype< _CharT > __ctype_type`
- typedef `basic_ios< _CharT, _Traits > __ios_type`
- typedef `basic_istream< char_type, traits_type > __istream_type`
- typedef `num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > __num_get_type`
- typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`
- typedef `basic_string< _CharT, _Traits, _Alloc > __string_type`
- typedef `basic_stringbuf< _CharT, _Traits, _Alloc > __stringbuf_type`
- typedef `_Alloc allocator_type`
- typedef `_CharT char_type`
- enum `event { erase_event, imbue_event, copyfmt_event }`
- typedef `void(* event_callback)(event, ios_base &, int)`
- typedef `_Ios_Fmtflags fmtflags`
- typedef `traits_type::int_type int_type`
- typedef `int io_state`

- typedef `_Ios_Iostate` [iostate](#)
  - typedef `traits_type::off_type` [off\\_type](#)
  - typedef `int` [open\\_mode](#)
  - typedef `_Ios_Openmode` [openmode](#)
  - typedef `traits_type::pos_type` [pos\\_type](#)
  - typedef `int` [seek\\_dir](#)
  - typedef `_Ios_Seekdir` [seekdir](#)
  - typedef `std::streamoff` [streamoff](#)
  - typedef `std::streampos` [streampos](#)
  - typedef `_Traits` [traits\\_type](#)
- 
- typedef `num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>>` [\\_\\_num\\_put\\_type](#)

#### Public Member Functions

- [basic\\_istream](#) ([ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#))
- [basic\\_istream](#) (const [\\_\\_string\\_type](#) &\_\_str, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#))
- [~basic\\_istream](#) ()
- const [locale](#) & [\\_M\\_getloc](#) () const
- void [\\_M\\_setstate](#) ([iostate](#) \_\_state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) \_\_state=[goodbit](#))
- [basic\\_ios](#) & [copyfmt](#) (const [basic\\_ios](#) &\_\_rhs)
- bool [eof](#) () const
- [iostate](#) [exceptions](#) () const
- void [exceptions](#) ([iostate](#) \_\_except)
- bool [fail](#) () const
- [char\\_type](#) [fill](#) () const
- [char\\_type](#) [fill](#) ([char\\_type](#) \_\_ch)
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) \_\_fmtfl)
- [streamsize](#) [gcount](#) () const
- template<>  
    [basic\\_istream](#)< [char](#) > & [getline](#) ([char\\_type](#) \*\_\_s, [streamsize](#) \_\_n, [char\\_type](#) \_\_delim)
- template<>  
    [basic\\_istream](#)< [wchar\\_t](#) > & [getline](#) ([char\\_type](#) \*\_\_s, [streamsize](#) \_\_n, [char\\_type](#) \_\_delim)
- [locale](#) [getloc](#) () const
- bool [good](#) () const

- `template<>`  
`basic_istream< char > & ignore (streamsize __n, int_type __delim)`
  - `template<>`  
`basic_istream< char > & ignore (streamsize __n)`
  - `template<>`  
`basic_istream< wchar_t > & ignore (streamsize __n)`
  - `template<>`  
`basic_istream< wchar_t > & ignore (streamsize __n, int_type __delim)`
  - `locale imbue (const locale &__loc)`
  - `long & iword (int __ix)`
  - `char narrow (char_type __c, char __dfault) const`
  - `streamsize precision () const`
  - `streamsize precision (streamsize __prec)`
  - `void *& pword (int __ix)`
  - `__stringbuf_type * rdbuf () const`
  - `basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > * __sb)`
  - `iosstate rdstate () const`
  - `void register_callback (event_callback __fn, int __index)`
  - `fmtflags setf (fmtflags __fmtfl)`
  - `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
  - `void setstate (iosstate __state)`
  - `void str (const __string_type &__s)`
  - `__string_type str () const`
  - `basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > * __tistr)`
  - `basic_ostream< _CharT, _Traits > * tie () const`
  - `void unsetf (fmtflags __mask)`
  - `char_type widen (char __c) const`
  - `streamsize width (streamsize __wide)`
  - `streamsize width () const`
- 
- `__istream_type & operator>> (__istream_type &(__pf)(__istream_type &))`
  - `__istream_type & operator>> (__ios_type &(__pf)(__ios_type &))`
  - `__istream_type & operator>> (ios_base &(__pf)(ios_base &))`

### Arithmetic Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to false. This has several effects, concluding with the setting of a status flag; see the `sentry` documentation for more.

*If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.*

*If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.*

- `__istream_type & operator>> (bool &__n)`
- `__istream_type & operator>> (short &__n)`
- `__istream_type & operator>> (unsigned short &__n)`
- `__istream_type & operator>> (int &__n)`
- `__istream_type & operator>> (unsigned int &__n)`
- `__istream_type & operator>> (long &__n)`
- `__istream_type & operator>> (unsigned long &__n)`
- `__istream_type & operator>> (long long &__n)`
- `__istream_type & operator>> (unsigned long long &__n)`
- `__istream_type & operator>> (float &__f)`
- `__istream_type & operator>> (double &__f)`
- `__istream_type & operator>> (long double &__f)`
- `__istream_type & operator>> (void *&__p)`
- `__istream_type & operator>> (__streambuf_type *__sb)`

### Unformatted Input Functions

*All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.*

*If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.*

*The number of characters extracted is stored for later retrieval by `gcount()`.*

*If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.*

- `int_type get ()`
- `__istream_type & get (char_type &__c)`
- `__istream_type & get (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & get (char_type *__s, streamsize __n)`
- `__istream_type & get (__streambuf_type &__sb, char_type __delim)`
- `__istream_type & get (__streambuf_type &__sb)`
- `__istream_type & getline (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & getline (char_type *__s, streamsize __n)`
- `__istream_type & ignore ()`
- `__istream_type & ignore (streamsize __n)`
- `__istream_type & ignore (streamsize __n, int_type __delim)`
- `int_type peek ()`
- `__istream_type & read (char_type *__s, streamsize __n)`

- [streamsize](#) [readsome](#) ([char\\_type](#) \*\_\_s, [streamsize](#) \_\_n)
- [\\_\\_istream\\_type](#) & [putback](#) ([char\\_type](#) \_\_c)
- [\\_\\_istream\\_type](#) & [unget](#) ()
- [int](#) [sync](#) ()
- [pos\\_type](#) [tellg](#) ()
- [\\_\\_istream\\_type](#) & [seekg](#) ([pos\\_type](#))
- [\\_\\_istream\\_type](#) & [seekg](#) ([off\\_type](#), [ios\\_base::seekdir](#))
  
- [operator void \\*](#) () const
- [bool](#) [operator!](#) () const

#### **Static Public Member Functions**

- [static bool](#) [sync\\_with\\_stdio](#) ([bool](#) \_\_sync=true)
- [static int](#) [xalloc](#) () [throw](#) ()

#### **Static Public Attributes**

- [static const](#) [fmtflags](#) [adjustfield](#)
- [static const](#) [openmode](#) [app](#)
- [static const](#) [openmode](#) [ate](#)
- [static const](#) [iostate](#) [badbit](#)
- [static const](#) [fmtflags](#) [basefield](#)
- [static const](#) [seekdir](#) [beg](#)
- [static const](#) [openmode](#) [binary](#)
- [static const](#) [fmtflags](#) [boolalpha](#)
- [static const](#) [seekdir](#) [cur](#)
- [static const](#) [fmtflags](#) [dec](#)
- [static const](#) [seekdir](#) [end](#)
- [static const](#) [iostate](#) [eofbit](#)
- [static const](#) [iostate](#) [failbit](#)
- [static const](#) [fmtflags](#) [fixed](#)
- [static const](#) [fmtflags](#) [floatfield](#)
- [static const](#) [iostate](#) [goodbit](#)
- [static const](#) [fmtflags](#) [hex](#)
- [static const](#) [openmode](#) [in](#)
- [static const](#) [fmtflags](#) [internal](#)
- [static const](#) [fmtflags](#) [left](#)
- [static const](#) [fmtflags](#) [oct](#)
- [static const](#) [openmode](#) [out](#)
- [static const](#) [fmtflags](#) [right](#)
- [static const](#) [fmtflags](#) [scientific](#)
- [static const](#) [fmtflags](#) [showbase](#)

- static const [fmtflags showpoint](#)
- static const [fmtflags showpos](#)
- static const [fmtflags skipws](#)
- static const [openmode trunc](#)
- static const [fmtflags unitbuf](#)
- static const [fmtflags uppercase](#)

### Protected Types

- enum { `_S_local_word_size` }

### Protected Member Functions

- void `_M_cache_locale` (const [locale](#) &\_\_loc)
- void `_M_call_callbacks` ([event](#) \_\_ev) throw ()
- void `_M_dispose_callbacks` (void) throw ()
- template<typename \_ValueT >  
  [\\_\\_istream\\_type](#) & `_M_extract` (\_ValueT &\_\_v)
- `_Words` & `_M_grow_words` (int \_\_index, bool \_\_iword)
- void `_M_init` () throw ()
- void `init` ([basic\\_streambuf](#)< \_CharT, \_Traits > \*\_\_sb)

### Protected Attributes

- `_Callback_list` \* `_M_callbacks`
- const [\\_\\_ctype\\_type](#) \* `_M_ctype`
- [iostate](#) `_M_exception`
- [char\\_type](#) `_M_fill`
- bool `_M_fill_init`
- [fmtflags](#) `_M_flags`
- [streamsize](#) `_M_gcount`
- [locale](#) `_M_ios_locale`
- `_Words` `_M_local_word` [`_S_local_word_size`]
- const [\\_\\_num\\_get\\_type](#) \* `_M_num_get`
- const [\\_\\_num\\_put\\_type](#) \* `_M_num_put`
- [streamsize](#) `_M_precision`
- [basic\\_streambuf](#)< \_CharT, \_Traits > \* `_M_streambuf`
- [iostate](#) `_M_streambuf_state`
- [basic\\_ostream](#)< \_CharT, \_Traits > \* `_M_tie`
- [streamsize](#) `_M_width`
- `_Words` \* `_M_word`
- int `_M_word_size`
- `_Words` `_M_word_zero`

## Friends

- class `sentry`

### 5.383.1 Detailed Description

`template<typename _CharT, typename _Traits, typename _Alloc> class std::basic_istringstream<_CharT, _Traits, _Alloc>`

Controlling input for `std::string`.

This class supports reading from objects of type `std::basic_string`, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

Definition at line 258 of file `sstream`.

### 5.383.2 Member Typedef Documentation

5.383.2.1 `template<typename _CharT, typename _Traits> typedef type<_CharT> std::basic_istream<_CharT, _Traits>::__ctype_type [inherited]`

These are non-standard types.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Definition at line 73 of file `istream`.

5.383.2.2 `template<typename _CharT, typename _Traits> typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits>> std::basic_istream<_CharT, _Traits>::__num_get_type [inherited]`

These are non-standard types.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Definition at line 72 of file `istream`.

5.383.2.3 `template<typename _CharT, typename _Traits> typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>> std::basic_ios<_CharT, _Traits>::__num_put_type [inherited]`

These are non-standard types.

### 5.383 `std::basic_istream< _CharT, _Traits, _Alloc >` Class Template Reference 1957

---

Reimplemented in [std::basic\\_ostream< \\_CharT, \\_Traits >](#), [std::basic\\_ostream< char, \\_Traits >](#), and [std::basic\\_ostream< char >](#).

Definition at line 86 of file `basic_ios.h`.

**5.383.2.4** `template<typename _CharT, typename _Traits, typename _Alloc>`  
`typedef _CharT std::basic_istream< _CharT, _Traits, _Alloc`  
`>::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_istream< \\_CharT, \\_Traits >](#).

Definition at line 262 of file `sstream`.

**5.383.2.5** `typedef void(* std::ios_base::event_callback)(event, ios_base &, int)`  
`[inherited]`

The type of an event callback function.

#### Parameters

*event* One of the members of the event enum.

*ios\_base* Reference to the [ios\\_base](#) object.

*int* The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several [ios\\_base](#) and [basic\\_ios](#) functions, specifically [imbue\(\)](#), [copyfmt\(\)](#), and [~ios\(\)](#).

Definition at line 438 of file `ios_base.h`.

**5.383.2.6** `typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]`

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`



- dec
- fixed
- hex
- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Definition at line 257 of file `ios_base.h`.

**5.383.2.7** `template<typename _CharT, typename _Traits, typename _Alloc>`  
`typedef traits_type::int_type std::basic_istream< _CharT,`  
`_Traits, _Alloc >::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_istream< \\_CharT, \\_Traits >](#).

Definition at line 267 of file `sstream`.

**5.383.2.8 typedef \_Ios\_Iostate std::ios\_base::iostate [inherited]**

This is a bitmask type.

*\_Ios\_Iostate* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type *iostate* are:

- badbit
- eofbit
- failbit
- goodbit

Definition at line 332 of file *ios\_base.h*.

**5.383.2.9 template<typename \_CharT, typename \_Traits, typename \_Alloc> typedef traits\_type::off\_type std::basic\_istream< \_CharT, \_Traits, \_Alloc >::off\_type**

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_istream< \\_CharT, \\_Traits >](#).

Definition at line 269 of file *sstream*.

**5.383.2.10 typedef \_Ios\_Openmode std::ios\_base::openmode [inherited]**

This is a bitmask type.

*\_Ios\_Openmode* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type *openmode* are:

- app
- ate
- binary
- in

- out
- trunc

Definition at line 363 of file ios\_base.h.

**5.383.2.11** `template<typename _CharT, typename _Traits, typename _Alloc>  
typedef traits_type::pos_type std::basic_istream< _CharT,  
_Traits, _Alloc >::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_istream< \\_CharT, \\_Traits >](#).

Definition at line 268 of file sstream.

**5.383.2.12** `typedef _Ios_Seekdir std::ios_base::seekdir [inherited]`

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- beg
- cur, equivalent to `SEEK_CUR` in the C standard library.
- end, equivalent to `SEEK_END` in the C standard library.

Definition at line 395 of file ios\_base.h.

**5.383.2.13** `template<typename _CharT, typename _Traits, typename _Alloc>  
typedef _Traits std::basic_istream< _CharT, _Traits, _Alloc  
>::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_istream< \\_CharT, \\_Traits >](#).

Definition at line 263 of file sstream.

### 5.383.3 Member Enumeration Documentation

#### 5.383.3.1 `enum std::ios_base::event` [inherited]

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 421 of file `ios_base.h`.

### 5.383.4 Constructor & Destructor Documentation

#### 5.383.4.1 `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_istream<_CharT, _Traits, _Alloc>::basic_istream ( ios_base::openmode __mode = ios_base::in ) [inline, explicit]`

Default constructor starts with an empty string buffer.

##### Parameters

*mode* Whether the buffer can read, or write, or both.

`ios_base::in` is automatically included in *mode*.

Initializes `sb` using `mode|in`, and passes `&sb` to the base class initializer. Does not allocate any buffer.

That's a lie. We initialize the base class with `NULL`, because the string class does its own memory management.

Definition at line 294 of file `sstream`.

References `std::basic_ios<_CharT, _Traits>::init()`.

#### 5.383.4.2 `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_istream<_CharT, _Traits, _Alloc>::basic_istream ( const __string_type & __str, ios_base::openmode __mode = ios_base::in ) [inline, explicit]`

Starts with an existing string buffer.

### Parameters

*str* A string to copy as a starting buffer.

*mode* Whether the buffer can read, or write, or both.

`ios_base::in` is automatically included in *mode*.

Initializes `sb` using *str* and `mode|in`, and passes `&sb` to the base class initializer.

That's a lie. We initialize the base class with `NULL`, because the string class does its own memory management.

Definition at line 312 of file `sstream`.

References `std::basic_ios< _CharT, _Traits >::init()`.

**5.383.4.3** `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_istream< _CharT, _Traits, _Alloc >::~basic_istream ( ) [inline]`

The destructor does nothing.

The buffer is deallocated by the `stringbuf` object, not the formatting stream.

Definition at line 323 of file `sstream`.

### 5.383.5 Member Function Documentation

**5.383.5.1** `const locale& std::ios_base::_M_getloc ( ) const [inline, inherited]`

Locale access.

### Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 708 of file `ios_base.h`.

Referenced by `std::money_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_time()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::time_put< _CharT, _OutIter >::do_put()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::time_put< _CharT, _OutIter >::put()`.

**5.383.5.2** `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::bad ( ) const` [**inline, inherited**]

Fast error checking.

#### Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 203 of file `basic_ios.h`.

**5.383.5.3** `template<typename _CharT, typename _Traits > void std::basic_ios< _CharT, _Traits >::clear ( iostate __state = goodbit )` [**inherited**]

[Re]sets the error state.

#### Parameters

*state* The new state flag(s) to set.

See [std::ios\\_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

Definition at line 42 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::exceptions()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::rdstate()`.

Referenced by `std::basic_ios< char, _Traits >::exceptions()`, `std::basic_fstream< _CharT, _Traits >::open()`, `std::basic_ofstream< _CharT, _Traits >::open()`, `std::basic_ifstream< _CharT, _Traits >::open()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ios< char, _Traits >::setstate()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

**5.383.5.4** `template<typename _CharT, typename _Traits > basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt ( const basic_ios< _CharT, _Traits > & __rhs )` [**inherited**]

Copies fields of `__rhs` into this.

#### Parameters

`__rhs` The source values for the copies.

#### Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy `exceptions()`.

Definition at line 64 of file `basic_ios.tcc`.

References `std::basic_ios< _CharT, _Traits >::exceptions()`, `std::basic_ios< _CharT, _Traits >::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios< _CharT, _Traits >::tie()`, and `std::ios_base::width()`.

#### 5.383.5.5 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::eof( ) const [inline, inherited]`

Fast error checking.

#### Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 182 of file `basic_ios.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

#### 5.383.5.6 `template<typename _CharT, typename _Traits> iostate std::basic_ios< _CharT, _Traits >::exceptions( ) const [inline, inherited]`

Throwing exceptions on errors.

### Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of [exceptions\(iostate\)](#) for the meaning of the return value.

Definition at line 214 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::clear()`, and `std::basic_ios<_CharT, _Traits>::copyfmt()`.

**5.383.5.7** `template<typename _CharT, typename _Traits> void std::basic_ios<_CharT, _Traits>::exceptions ( iostate __except ) [inline, inherited]`

Throwing exceptions on errors.

### Parameters

*except* The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type [std::ios\\_base::failure](#) is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
 std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

 std::ifstream f ("/etc/motd");

 std::cerr << "Setting badbit\n";
 f.setstate (std::ios_base::badbit);

 std::cerr << "Setting exception mask\n";
 f.exceptions (std::ios_base::badbit);
}
```

Definition at line 249 of file `basic_ios.h`.



**5.383.5.8** `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits >::fail( ) const [inline, inherited]`

Fast error checking.

#### Returns

True if either the badbit or the failbit is set.

Checking the badbit in [fail\(\)](#) is historical practice. Note that other iostate flags may also be set.

Definition at line 193 of file basic\_ios.h.

Referenced by `std::basic_ios< char, _Traits >::operator void *()`, `std::basic_ios< char, _Traits >::operator!()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::regex_traits< _Ch_type >::value()`.

**5.383.5.9** `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill( ) const [inline, inherited]`

Retrieves the *empty* character.

#### Returns

The current fill character.

It defaults to a space ( ' ' ) in the current locale.

Definition at line 362 of file basic\_ios.h.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, and `std::basic_ios< char, _Traits >::fill()`.

**5.383.5.10** `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill( char_type __ch ) [inline, inherited]`

Sets a new *empty* character.

**Parameters**

*ch* The new character.

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 382 of file `basic_ios.h`.

**5.383.5.11 fmtflags std::ios\_base::flags ( ) const [inline, inherited]**

Access to format flags.

**Returns**

The format control flags for both input and output.

Definition at line 553 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, `std::operator>>()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

**5.383.5.12 fmtflags std::ios\_base::flags ( fmtflags \_\_fmtfl ) [inline, inherited]**

Setting new format flags all at once.

**Parameters**

*fmtfl* The new flags to set.

**Returns**

The previous format control flags.

This function overwrites all the format flags with *fmtfl*.

Definition at line 564 of file `ios_base.h`.

**5.383.5.13** `template<typename _CharT, typename _Traits> streamsize  
std::basic_istream<_CharT, _Traits>::gcount ( ) const  
[inline, inherited]`

Character counting.

**Returns**

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 251 of file istream.

**5.383.5.14** `template<typename _CharT, typename _Traits> basic_istream<  
_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits  
>::get ( void ) [inherited]`

Simple extraction.

**Returns**

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 238 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.383.5.15** `template<typename _CharT, typename _Traits> basic_istream<  
_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get (   
char_type & __c ) [inherited]`

Simple extraction.

**Parameters**

*c* The character in which to store data.

### Returns

\*this

Tries to extract a character and store it in *c*. If none are available, sets failbit and returns traits::eof().

### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 274 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits >::\_M\_gcount, std::ios\_base::badbit, std::ios\_base::eofbit, std::ios\_base::failbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), and std::basic\_ios< \_CharT, \_Traits >::setstate().

**5.383.5.16** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::get ( char_type * __s, streamsize __n ) [inline, inherited]`

Simple multiple-character extraction.

### Parameters

*s* Pointer to an array.

*n* Maximum number of characters to store in *s*.

### Returns

\*this

Returns get(s,n,widen('\n')).

Definition at line 335 of file istream.

**5.383.5.17** `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get ( __streambuf_type & __sb, char_type __delim ) [inherited]`

Extraction into another streambuf.

### Parameters

*sb* A streambuf in which to store data.

*delim* A "stop" character.

#### Returns

\*this

Characters are extracted and inserted into *sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 358 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits >::\_M\_gcount, std::ios\_base::badbit, std::basic\_ios< \_CharT, \_Traits >::eof(), std::ios\_base::eofbit, std::ios\_base::failbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), std::basic\_ios< \_CharT, \_Traits >::setstate(), std::basic\_streambuf< \_CharT, \_Traits >::sgetc(), std::basic\_streambuf< \_CharT, \_Traits >::snextc(), and std::basic\_streambuf< \_CharT, \_Traits >::sputc().

**5.383.5.18** `template<typename _CharT, typename _Traits> basic_istream<  
_CharT, _Traits> & std::basic_istream< _CharT, _Traits>::get  
( char_type * __s, streamsize __n, char_type __delim )  
[inherited]`

Simple multiple-character extraction.

#### Parameters

*s* Pointer to an array.

*n* Maximum number of characters to store in *s*.

*delim* A "stop" character.

#### Returns

\*this

Characters are extracted and stored into *s* until one of the following happens:

- $n-1$  characters are stored
- the input sequence reaches EOF
- the next character equals *delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 311 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

**5.383.5.19** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::get ( __streambuf_type & __sb ) [inline, inherited]`

Extraction into another streambuf.

#### Parameters

*sb* A streambuf in which to store data.

#### Returns

\*this

Returns `get(sb,widen('\n'))`.

Definition at line 368 of file istream.

**5.383.5.20** `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::getline ( char_type * __s, streamsize __n, char_type __delim ) [inherited]`

String extraction.

### Parameters

- s* A character array in which to store the data.
- n* Maximum number of characters to extract.
- delim* A "stop" character.

### Returns

\*this

Extracts and stores characters into *s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals *delim*, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. *n*−1 characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 402 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_streambuf< _CharT, _Traits >::sbumpc()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snexctc()`.

**5.383.5.21** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::getline ( char_type * __s, streamsize __n ) [inline, inherited]`

String extraction.

### Parameters

- s* A character array in which to store the data.
- n* Maximum number of characters to extract.

**Returns**

\*this

Returns `getline(s,n,widen('\n'))`.

Definition at line 408 of file `istream`.

Referenced by `std::basic_istream< char >::getline()`.

**5.383.5.22 locale std::ios\_base::getloc ( ) const [inline, inherited]**

Locale access.

**Returns**

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 697 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::money_put< _CharT, _OutputIter >::do_put()`, `std::basic_ios< _CharT, _Traits >::imbue()`, `std::operator>>()`, and `std::ws()`.

**5.383.5.23 template<typename \_CharT, typename \_Traits> bool  
std::basic\_ios< \_CharT, \_Traits >::good ( ) const [inline,  
inherited]**

Fast error checking.

**Returns**

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 172 of file `basic_ios.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.



**5.383.5.24** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore( streamsize __n ) [inherited]`

Simple extraction.

#### Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 495 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

**5.383.5.25** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore( void ) [inherited]`

Discarding characters.

#### Parameters

*n* Number of characters to discard.

*delim* A "stop" character.

#### Returns

`*this`

Extracts characters and throws them away until one of the following happens:

- if *n* != `std::numeric_limits<int>::max()`, *n* characters are extracted
- the input sequence reaches end-of-file
- the next character equals *delim* (in this case, the character is extracted); note that this condition will never occur if *delim* equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 462 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits >::\_M\_gcount, std::ios\_base::badbit, std::basic\_ios< \_CharT, \_Traits >::eof(), std::ios\_base::eofbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), std::basic\_streambuf< \_CharT, \_Traits >::sbumpc(), and std::basic\_ios< \_CharT, \_Traits >::setstate().

**5.383.5.26** `template<typename _CharT, typename _Traits > basic_istream<  
_CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore  
( streamsize __n, int_type __delim ) [inherited]`

Simple extraction.

#### Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 557 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits >::\_M\_gcount, std::ios\_base::badbit, std::basic\_ios< \_CharT, \_Traits >::eof(), std::ios\_base::eofbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), std::basic\_streambuf< \_CharT, \_Traits >::sbumpc(), std::basic\_ios< \_CharT, \_Traits >::setstate(), std::basic\_streambuf< \_CharT, \_Traits >::sgetc(), and std::basic\_streambuf< \_CharT, \_Traits >::nextc().

**5.383.5.27** `template<typename _CharT, typename _Traits > locale  
std::basic_ios< _CharT, _Traits >::imbue ( const locale & __loc )  
[inherited]`

Moves to a new locale.

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

### 5.383 `std::basic_istream< _CharT, _Traits, _Alloc >` Class Template Reference 1976

---

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Reimplemented from `std::ios_base`.

Definition at line 115 of file `basic_ios.tcc`.

References `std::ios_base::getloc()`, and `std::basic_ios< _CharT, _Traits >::rdbuf()`.

Referenced by `std::operator<<()`.

**5.383.5.28** `template<typename _CharT, typename _Traits> void  
std::basic_ios< _CharT, _Traits >::init ( basic_streambuf<  
_CharT, _Traits > * __sb ) [protected, inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 127 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

Referenced by `std::basic_fstream< _CharT, _Traits >::basic_fstream()`, `std::basic_ifstream< _CharT, _Traits >::basic_ifstream()`, `std::basic_ios< char, _Traits >::basic_ios()`, `std::basic_istream< char >::basic_istream()`, `std::basic_istreamstream< _CharT, _Traits, _Alloc >::basic_istreamstream()`, `std::basic_ofstream< _CharT, _Traits >::basic_ofstream()`, `std::basic_ostream< char >::basic_ostream()`, `std::basic_ostringstream< _CharT, _Traits, _Alloc >::basic_ostringstream()`, and `std::basic_stringstream< _CharT, _Traits, _Alloc >::basic_stringstream()`.

**5.383.5.29** `long& std::ios_base::iword ( int __ix ) [inline, inherited]`

Access to integer array.

#### Parameters

`__ix` Index into the array.

#### Returns

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any

### 5.383 `std::basic_istream<_CharT, _Traits, _Alloc>` Class Template Reference 1977

---

purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 743 of file `ios_base.h`.

**5.383.5.30** `template<typename _CharT, typename _Traits> char  
std::basic_ios<_CharT, _Traits>::narrow ( char_type __c, char  
__dfault ) const [inline, inherited]`

Squeezes characters.

#### Parameters

*c* The character to narrow.  
*dfault* The character to narrow.

#### Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, dfault)
```

Additional notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 422 of file `basic_ios.h`.

**5.383.5.31** `template<typename _CharT, typename _Traits> std::basic_ios<  
_CharT, _Traits>::operator void * ( ) const [inline,  
inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 113 of file `basic_ios.h`.

**5.383.5.32** `template<typename _CharT, typename _Traits> bool  
std::basic_ios<_CharT, _Traits>::operator! ( ) const  
[inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 117 of file `basic_ios.h`.

**5.383.5.33** `template<typename _CharT, typename _Traits> basic_istream<  
_CharT, _Traits> & std::basic_istream<_CharT, _Traits  
>::operator>> ( short & __n ) [inherited]`

Basic arithmetic extractors.

#### Parameters

`A` variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 116 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get<_CharT, _InIter>::get()`, `std::ios_base::goodbit`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.383.5.34** `template<typename _CharT, typename _Traits> __istream_type&  
std::basic_istream<_CharT, _Traits>::operator>> ( float & __f  
) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

`A` variable of builtin type.

### 5.383 `std::basic_istream<_CharT, _Traits, _Alloc>` Class Template Reference 1979

---

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 205 of file `istream`.

**5.383.5.35** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( long double & __f ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

`A` variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 213 of file `istream`.

**5.383.5.36** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( unsigned short & __n ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

`A` variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 176 of file `istream`.

**5.383** `std::basic_istream<_CharT, _Traits, _Alloc>` Class Template Reference 1980

---

**5.383.5.37** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>>( void *& __p ) [inline, inherited]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 217 of file `istream`.

**5.383.5.38** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::operator>>( int & __n ) [inherited]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 161 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get<_CharT, _InIter>::get()`, `std::ios_base::goodbit`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.383.5.39** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::operator>>( __streambuf_type * __sb ) [inherited]`

Extracting into another streambuf.

#### Parameters

*sb* A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the *sb* streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 206 of file istream.tcc.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.383.5.40** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( unsigned int & __n ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 183 of file istream.



**5.383.5.41** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>> ( __istream_type &(*)(__istream_type &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iomanip` header.

Definition at line 122 of file `istream`.

**5.383.5.42** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>> ( __ios_type &(*)(__ios_type &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iomanip` header.

Definition at line 126 of file `istream`.

**5.383.5.43** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>> ( ios_base &(*)(ios_base &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iomanip` header.

Definition at line 133 of file `istream`.

**5.383.5.44** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>> ( long & __n ) [inline, inherited]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 187 of file `istream`.

**5.383.5.45** `template<typename _CharT, typename _Traits> __istream_type&  
std::basic_istream<_CharT, _Traits >::operator>> ( unsigned  
long & __n ) [inline, inherited]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 191 of file `istream`.

**5.383.5.46** `template<typename _CharT, typename _Traits> __istream_type&  
std::basic_istream<_CharT, _Traits >::operator>> ( double &  
__f ) [inline, inherited]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

`*this` if successful

### 5.383 `std::basic_istream<_CharT, _Traits, _Alloc>` Class Template Reference 1984

---

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 209 of file `istream`.

**5.383.5.47** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( long long & __n ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 196 of file `istream`.

**5.383.5.48** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( bool & __n ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 169 of file `istream`.

**5.383.5.49** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( unsigned long long & __n ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

`A` variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 200 of file `istream`.

**5.383.5.50** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits>::peek ( void ) [inherited]`

Looking ahead in the stream.

#### Returns

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 622 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.383.5.51** `streamsize std::ios_base::precision ( ) const [inline, inherited]`

Flags access.

**Returns**

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 623 of file ios\_base.h.

Referenced by std::basic\_ios< \_CharT, \_Traits >::copyfmt(), and std::operator<<().

**5.383.5.52 streamsize std::ios\_base::precision ( streamsize \_\_prec )**  
**[inline, inherited]**

Changing flags.

**Parameters**

*prec* The new precision value.

**Returns**

The previous value of [precision\(\)](#).

Definition at line 632 of file ios\_base.h.

**5.383.5.53 template<typename \_CharT, typename \_Traits > basic\_istream<**  
**\_CharT, \_Traits > & std::basic\_istream< \_CharT, \_Traits**  
**>::putback ( char\_type \_\_c ) [inherited]**

Unextracting a single character.

**Parameters**

*c* The character to push back into the input stream.

**Returns**

\*this

If [rdbuf\(\)](#) is not null, calls [rdbuf\(\)->sputbackc\(c\)](#).

If [rdbuf\(\)](#) is null or if [sputbackc\(\)](#) fails, sets badbit in the error state.

**Note**

Since no characters are extracted, the next call to [gcount\(\)](#) will return 0, as required by DR 60.

Definition at line 713 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits >::\_M\_gcount, std::ios\_base::badbit, std::basic\_ios< \_CharT, \_Traits >::clear(), std::basic\_ios< \_CharT, \_Traits >::eof(), std::ios\_base::eofbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), std::basic\_ios< \_CharT, \_Traits >::rdstate(), std::basic\_ios< \_CharT, \_Traits >::setstate(), and std::basic\_streambuf< \_CharT, \_Traits >::sputbackc().

Referenced by std::operator>>().

**5.383.5.54 void\*& std::ios\_base::pword ( int \_\_ix ) [inline, inherited]**

Access to void pointer array.

**Parameters**

`__ix` Index into the array.

**Returns**

A reference to a void\* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 764 of file ios\_base.h.

**5.383.5.55 template<typename \_CharT, typename \_Traits, typename \_Alloc> \_\_stringbuf\_type\* std::basic\_istream< \_CharT, \_Traits, \_Alloc >::rdbuf ( ) const [inline]**

Accessing the underlying buffer.

**Returns**

The current `basic_stringbuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Definition at line 334 of file `sstream`.

**5.383.5.56** `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rdbuf ( basic_streambuf<_CharT, _Traits> * __sb ) [inherited]`

Changing the underlying buffer.

#### Parameters

*sb* The new stream buffer.

#### Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream foo; // or some other derived type
std::streambuf* p =;

foo.ios::rdbuf(p); // ios == basic_ios<char>
```

Definition at line 54 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::clear()`.

**5.383.5.57** `template<typename _CharT, typename _Traits> iostate std::basic_ios<_CharT, _Traits>::rdstate ( ) const [inline, inherited]`

Returns the error state of the stream buffer.

#### Returns

A bit pattern (well, isn't everything?)

See [std::ios\\_base::iostate](#) for the possible bit values. Most users will call one of the interpreting wrappers, e.g., [good\(\)](#).

Definition at line 129 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::basic_ios< char, _Traits >::eof()`, `std::basic_ios< char, _Traits >::fail()`, `std::basic_ios< char, _Traits >::good()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ios< char, _Traits >::setstate()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

**5.383.5.58** `template<typename _CharT, typename _Traits> basic_istream<  
_CharT, _Traits> & std::basic_istream< _CharT, _Traits>::read (  
char_type * __s, streamsize __n ) [inherited]`

Extraction without delimiters.

#### Parameters

- s* A character array.
- n* Maximum number of characters to store.

#### Returns

\*this

If the stream state is [good\(\)](#), extracts characters and stores them into *s* until one of the following happens:

- n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 652 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.



**5.383.5.59** `template<typename _CharT, typename _Traits> streamsize  
std::basic_istream<_CharT, _Traits>::readsome ( char_type *  
__s, streamsize __n ) [inherited]`

Extraction until the buffer is exhausted, but no more.

#### Parameters

- s* A character array.
- n* Maximum number of characters to store.

#### Returns

The number of characters extracted.

Extracts characters and stores them into *s* depending on the number of characters remaining in the streambuf's buffer, `rddbuf() -> in_avail()`, called *A* here:

- if *A* == -1, sets eofbit and extracts no characters
- if *A* == 0, extracts no characters
- if *A* > 0, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 681 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::min()`, `std::basic_ios<_CharT, _Traits>::rddbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.383.5.60** `void std::ios_base::register_callback ( event_callback __fn, int  
__index ) [inherited]`

Add the callback *\_\_fn* with parameter *\_\_index*.

#### Parameters

- \_\_fn* The function to add.
- \_\_index* The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**5.383.5.61** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg ( off_type __off, ios_base::seekdir __dir ) [inherited]`

Changing the current read position.

#### Parameters

*off* A file offset object.  
*dir* The direction in which to seek.

#### Returns

\*this

If `fail()` is not true, calls `rdbuf()` -> `pubseekoff(off, dir)`. If that function fails, sets failbit.

#### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 886 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::ios_base::eofbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::rdstate()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.383.5.62** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg ( pos_type __pos ) [inherited]`

Changing the current read position.

#### Parameters

*pos* A file position object.

#### Returns

\*this

### 5.383 `std::basic_istream< _CharT, _Traits, _Alloc >` Class Template Reference 1992

---

If `fail()` is not true, calls `rdbuf()` -> `pubseekpos(pos)`. If that function fails, sets failbit.

#### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 847 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::ios_base::eofbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::rdstate()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

#### 5.383.5.63 `fmtflags std::ios_base::setf( fmtflags __fmtfl, fmtflags __mask )` [inline, inherited]

Setting new format flags.

#### Parameters

*fmtfl* Additional flags to set.  
*mask* The flags mask for *fmtfl*.

#### Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 597 of file `ios_base.h`.

#### 5.383.5.64 `fmtflags std::ios_base::setf( fmtflags __fmtfl )` [inline, inherited]

Setting new format flags.

#### Parameters

*fmtfl* Additional flags to set.

### Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 580 of file ios\_base.h.

Referenced by std::dec(), std::fixed(), std::hex(), std::left(), std::oct(), std::right(), std::scientific(), std::showbase(), std::showpoint(), std::showpos(), std::skipws(), std::unitbuf(), and std::uppercase().

**5.383.5.65** `template<typename _CharT, typename _Traits> void  
std::basic_ios< _CharT, _Traits >::setstate ( iostate __state )  
[inline, inherited]`

Sets additional flags in the error state.

### Parameters

*state* The additional state flag(s) to set.

See [std::ios\\_base::iostate](#) for the possible bit values.

Definition at line 149 of file basic\_ios.h.

Referenced by std::basic\_ostream< char >::\_M\_write(), std::basic\_fstream< \_CharT, \_Traits >::close(), std::basic\_ofstream< \_CharT, \_Traits >::close(), std::basic\_ifstream< \_CharT, \_Traits >::close(), std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_fstream< \_CharT, \_Traits >::open(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::sync(), std::basic\_istream< \_CharT, \_Traits >::unget(), and std::ws().

**5.383.5.66** `template<typename _CharT, typename _Traits, typename _Alloc>  
void std::basic_istream< _CharT, _Traits, _Alloc >::str (   
const __string_type & __s ) [inline]`

Setting a new buffer.

#### Parameters

*s* The string to use as a new sequence.

Calls `rdbuf() -> str(s)`.

Definition at line 352 of file sstream.

**5.383.5.67** `template<typename _CharT, typename _Traits, typename _Alloc>  
__string_type std::basic_istream< _CharT, _Traits, _Alloc  
>::str ( ) const [inline]`

Copying out the string buffer.

#### Returns

`rdbuf() -> str()`

Definition at line 342 of file sstream.

**5.383.5.68** `template<typename _CharT, typename _Traits > int  
std::basic_istream< _CharT, _Traits >::sync ( void )  
[inherited]`

Synchronizing the stream buffer.

#### Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf() -> pubsync()`, and if that returns -1, sets badbit and returns -1.

Otherwise, returns 0.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 783 of file istream.tcc.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_streambuf< _CharT, _Traits >::pubsync()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.383.5.69 static bool std::ios\_base::sync\_with\_stdio ( bool \_\_sync = true )**  
**[static, inherited]**

Interaction with the standard C I/O objects.

**Parameters**

*sync* Whether to synchronize or not.

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

**5.383.5.70 template<typename \_CharT, typename \_Traits > basic\_istream< \_CharT, \_Traits >::pos\_type std::basic\_istream< \_CharT, \_Traits >::tell ( void ) [inherited]**

Getting the current read position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, in)`.

#### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 819 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::in`, and `std::basic_ios< _CharT, _Traits >::rdbuf()`.

**5.383.5.71** `template<typename _CharT, typename _Traits>`  
`basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits`  
`>::tie( ) const [inline, inherited]`

Fetches the current *tied* stream.

#### Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 287 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

**5.383.5.72** `template<typename _CharT, typename _Traits>`  
`basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits`  
`>::tie( basic_ostream< _CharT, _Traits > * __tiestr ) [inline,`  
`inherited]`

Ties this stream to an output stream.

#### Parameters

*tiestr* The output stream.

#### Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see [tie\(\)](#) for more.

Definition at line 299 of file `basic_ios.h`.

**5.383.5.73** `template<typename _CharT, typename _Traits> basic_istream<  
_CharT, _Traits> & std::basic_istream< _CharT, _Traits>::unget  
( void ) [inherited]`

Unextracting the previous character.

#### Returns

`*this`

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

#### Note

Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 748 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits>::M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits>::clear()`, `std::basic_ios< _CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits>::rdbuf()`, `std::basic_ios< _CharT, _Traits>::rdstate()`, `std::basic_ios< _CharT, _Traits>::setstate()`, and `std::basic_streambuf< _CharT, _Traits>::sungetc()`.

**5.383.5.74** `void std::ios_base::unsetf( fmtflags __mask ) [inline,  
inherited]`

Clearing format flags.

#### Parameters

*mask* The flags to unset.

This function clears *mask* in the format flags.

Definition at line 612 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.



**5.383.5.75** `template<typename _CharT, typename _Traits> char_type  
std::basic_ios<_CharT, _Traits>::widen ( char __c ) const  
[inline, inherited]`

Widens characters.

#### Parameters

*c* The character to widen.

#### Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 441 of file `basic_ios.h`.

Referenced by `std::endl()`, `std::basic_ios<char, _Traits>::fill()`, `std::basic_istream<char>::get()`, `std::basic_istream<char>::getline()`, `std::getline()`, and `std::operator>>()`.

**5.383.5.76** `streamsize std::ios_base::width ( ) const [inline,  
inherited]`

Flags access.

#### Returns

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 646 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::operator>>()`.

**5.383.5.77** `streamsize std::ios_base::width ( streamsize __wide ) [inline, inherited]`

Changing flags.

#### Parameters

*wide* The new width value.

#### Returns

The previous value of [width\(\)](#).

Definition at line 655 of file ios\_base.h.

**5.383.5.78** `static int std::ios_base::xalloc ( ) throw () [static, inherited]`

Access to unique indices.

#### Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

### 5.383.6 Member Data Documentation

**5.383.6.1** `template<typename _CharT, typename _Traits> streamsize std::basic_istream< _CharT, _Traits >::M_gcount [protected, inherited]`

The number of characters extracted in the previous unformatted function; see [gcount\(\)](#).

Definition at line 81 of file istream.

### 5.383 `std::basic_istream< _CharT, _Traits, _Alloc >` Class Template Reference 2000

---

Referenced by `std::basic_istream< char >::gcount()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::basic_istream< char >::~~basic_istream()`.

#### 5.383.6.2 `const fmtflags std::ios_base::adjustfield` **[static, inherited]**

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 312 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

#### 5.383.6.3 `const openmode std::ios_base::app` **[static, inherited]**

Seek to end before each write.

Definition at line 366 of file `ios_base.h`.

#### 5.383.6.4 `const openmode std::ios_base::ate` **[static, inherited]**

Open and seek to end immediately after opening.

Definition at line 369 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

#### 5.383.6.5 `const iostate std::ios_base::badbit` **[static, inherited]**

Indicates a loss of integrity in an input or output sequence (such /// as an irrecoverable read error from a file).

Definition at line 336 of file `ios_base.h`.

Referenced by `std::basic_ostream< char >::_M_write()`, `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::basic_ios< char,`

`_Traits >::fail()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ios< _CharT, _Traits >::init()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, `std::basic_istream< _CharT, _Traits >::unget()`, `std::basic_ostream< _CharT, _Traits >::write()`, and `std::basic_ostream< _CharT, _Traits >::sentry::~sentry()`.

#### **5.383.6.6 const fmtflags std::ios\_base::basefield [static, inherited]**

A mask of `dec|oct|hex`. Useful for the 2-arg form of `setf`.

Definition at line 315 of file `ios_base.h`.

Referenced by `std::dec()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::hex()`, `std::oct()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

#### **5.383.6.7 const seekdir std::ios\_base::beg [static, inherited]**

Request a seek relative to the beginning of the stream.

Definition at line 398 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::seekpos()`.

#### **5.383.6.8 const openmode std::ios\_base::binary [static, inherited]**

Perform input and output in binary mode (as opposed to text mode). `/// This is probably not what you think it is; see http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html.`

Definition at line 374 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::showmanyc()`.

**5.383.6.9 const fmtflags std::ios\_base::boolalpha [static, inherited]**

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 260 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::noboolalpha()`.

**5.383.6.10 const seekdir std::ios\_base::cur [static, inherited]**

Request a seek relative to the current position within the sequence.

Definition at line 401 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_filebuf< _CharT, _Traits >::seekoff()`, `std::basic_istream< _CharT, _Traits >::tellg()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

**5.383.6.11 const fmtflags std::ios\_base::dec [static, inherited]**

Converts integer input or generates integer output in decimal base.

Definition at line 263 of file `ios_base.h`.

Referenced by `std::dec()`.

**5.383.6.12 const seekdir std::ios\_base::end [static, inherited]**

Request a seek relative to the current end of the sequence.

Definition at line 404 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`.

**5.383.6.13 const iostate std::ios\_base::eofbit [static, inherited]**

Indicates that an input operation reached the end of an input sequence.

Definition at line 339 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_date(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::eof(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::unget(), and std::ws().

#### **5.383.6.14 const iostate std::ios\_base::failbit [static, inherited]**

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 344 of file ios\_base.h.

Referenced by std::basic\_fstream< \_CharT, \_Traits >::close(), std::basic\_ofstream< \_CharT, \_Traits >::close(), std::basic\_ifstream< \_CharT, \_Traits >::close(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::fail(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_fstream< \_CharT, \_Traits >::open(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry(), and std::basic\_istream< \_CharT, \_Traits >::sentry::sentry().

#### **5.383.6.15 const fmtflags std::ios\_base::fixed [static, inherited]**

Generate floating-point output in fixed-point notation.

Definition at line 266 of file ios\_base.h.

Referenced by std::fixed().

**5.383.6.16 const fmtflags std::ios\_base::floatfield [static, inherited]**

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 318 of file `ios_base.h`.

Referenced by std::fixed(), and std::scientific().

**5.383.6.17 const iostate std::ios\_base::goodbit [static, inherited]**

Indicates all is well.

Definition at line 347 of file `ios_base.h`.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_ios< \_CharT, \_Traits >::init(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::sync(), and std::basic\_istream< \_CharT, \_Traits >::unget().

**5.383.6.18 const fmtflags std::ios\_base::hex [static, inherited]**

Converts integer input or generates integer output in hexadecimal base.

Definition at line 269 of file `ios_base.h`.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), std::hex(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

**5.383.6.19 const openmode std::ios\_base::in [static, inherited]**

Open for input. Default for `ifstream` and `fstream`.

Definition at line 377 of file `ios_base.h`.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_set_buffer()`, `std::basic_ifstream< _CharT, _Traits >::open()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

**5.383.6.20 const fmtflags std::ios\_base::internal [static, inherited]**

Adds fill characters at a designated internal point in certain `///` generated output, or identical to `right` if no such point is `///` designated.

Definition at line 274 of file `ios_base.h`.

Referenced by `std::internal()`.

**5.383.6.21 const fmtflags std::ios\_base::left [static, inherited]**

Adds fill characters on the right (final positions) of certain `///` generated output. (I.e., the thing you print is flush left.).

Definition at line 278 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, and `std::left()`.

**5.383.6.22 const fmtflags std::ios\_base::oct [static, inherited]**

Converts integer input or generates integer output in octal base.

Definition at line 281 of file `ios_base.h`.

Referenced by `std::oct()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.



**5.383.6.23 const openmode std::ios\_base::out [static, inherited]**

Open for output. Default for `ofstream` and `fstream`.

Definition at line 380 of file `ios_base.h`.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_set_buffer()`, `std::basic_ofstream< _CharT, _Traits >::open()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

**5.383.6.24 const fmtflags std::ios\_base::right [static, inherited]**

Adds fill characters on the left (initial positions) of certain /// generated output. (I.e., the thing you print is flush right.).

Definition at line 285 of file `ios_base.h`.

Referenced by `std::right()`.

**5.383.6.25 const fmtflags std::ios\_base::scientific [static, inherited]**

Generates floating-point output in scientific notation.

Definition at line 288 of file `ios_base.h`.

Referenced by `std::scientific()`.

**5.383.6.26 const fmtflags std::ios\_base::showbase [static, inherited]**

Generates a prefix indicating the numeric base of generated integer /// output.

Definition at line 292 of file `ios_base.h`.

Referenced by `std::noshowbase()`, and `std::showbase()`.

**5.383.6.27 const fmtflags std::ios\_base::showpoint [static, inherited]**

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 296 of file ios\_base.h.

Referenced by std::noshowpoint(), and std::showpoint().

**5.383.6.28 const fmtflags std::ios\_base::showpos [static, inherited]**

Generates a + sign in non-negative generated numeric output.

Definition at line 299 of file ios\_base.h.

Referenced by std::noshowpos(), and std::showpos().

**5.383.6.29 const fmtflags std::ios\_base::skipws [static, inherited]**

Skips leading white space before certain input operations.

Definition at line 302 of file ios\_base.h.

Referenced by std::noskipws(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), and std::skipws().

**5.383.6.30 const openmode std::ios\_base::trunc [static, inherited]**

Open for input. Default for ofstream.

Definition at line 383 of file ios\_base.h.

**5.383.6.31 const fmtflags std::ios\_base::unitbuf [static, inherited]**

Flushes output after each output operation.

Definition at line 305 of file ios\_base.h.

Referenced by std::nounitbuf(), std::unitbuf(), and std::basic\_ostream< \_CharT, \_Traits >::sentry::~sentry().

**5.383.6.32 const fmtflags std::ios\_base::uppercase [static, inherited]**

Replaces certain lowercase letters with their uppercase equivalents /// in generated output.

Definition at line 309 of file ios\_base.h.

Referenced by std::num\_put< \_CharT, \_OutIter >::do\_put(), std::nouppercase(), and std::uppercase().

The documentation for this class was generated from the following file:

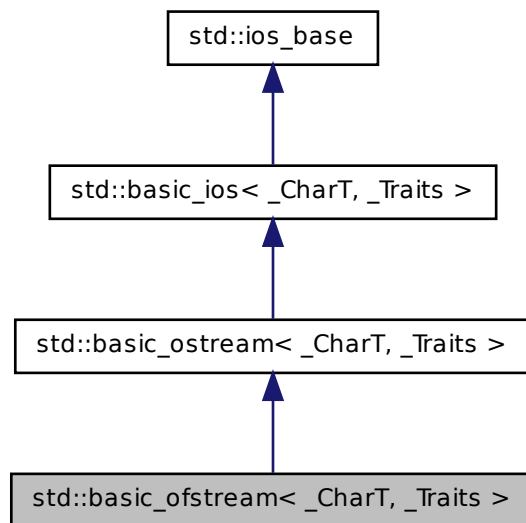
- [sstream](#)

**5.384 std::basic\_ofstream< \_CharT, \_Traits > Class Template Reference**

Controlling output for files.

This class supports reading from named files, using the inherited functions from [std::basic\\_ostream](#). To control the associated sequence, an instance of [std::basic\\_filebuf](#) is used, which this page refers to as `sb`.

Inheritance diagram for `std::basic_ofstream<_CharT, _Traits>`:



### Public Types

- typedef `ctype<_CharT>` `__ctype_type`
- typedef `basic_filebuf<char_type, traits_type>` `__filebuf_type`
- typedef `basic_ios<_CharT, _Traits>` `__ios_type`
- typedef `num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>>` `__num_put_type`
- typedef `basic_ostream<char_type, traits_type>` `__ostream_type`
- typedef `basic_streambuf<_CharT, _Traits>` `__streambuf_type`
- typedef `_CharT` `char_type`
- enum `event` { `erase_event`, `imbue_event`, `copyfmt_event` }
- typedef `void(* event_callback)(event, ios_base &, int)`
- typedef `_Ios_Fmtflags` `fmtflags`
- typedef `traits_type::int_type` `int_type`
- typedef `int` `io_state`
- typedef `_Ios_Iostate` `iostate`
- typedef `traits_type::off_type` `off_type`

- typedef int **open\_mode**
  - typedef `_Ios_Openmode` **openmode**
  - typedef `traits_type::pos_type` **pos\_type**
  - typedef int **seek\_dir**
  - typedef `_Ios_Seekdir` **seekdir**
  - typedef `std::streamoff` **streamoff**
  - typedef `std::streampos` **streampos**
  - typedef `_Traits` **traits\_type**
- 
- typedef `num_get<_CharT, istreambuf_iterator<_CharT, _Traits>>` **num\_get\_type**

### Public Member Functions

- `basic_ofstream()`
- `basic_ofstream (const char *__s, ios_base::openmode __mode=ios_base::out|ios_base::trunc)`
- `basic_ofstream (const std::string &__s, ios_base::openmode __mode=ios_base::out|ios_base::trunc)`
- `~basic_ofstream()`
- `const locale &_M_getloc() const`
- `void _M_setstate(iostate __state)`
- `bool bad() const`
- `void clear(iostate __state=goodbit)`
- `void close()`
- `basic_ios &copyfmt (const basic_ios &__rhs)`
- `bool eof() const`
- `iostate exceptions() const`
- `void exceptions(iostate __except)`
- `bool fail() const`
- `char_type fill() const`
- `char_type fill(char_type __ch)`
- `fmtflags flags() const`
- `fmtflags flags(fmtflags __fmtfl)`
- `__ostream_type &flush()`
- `locale getloc() const`
- `bool good() const`
- `locale imbue (const locale &__loc)`
- `bool is_open()`
- `bool is_open() const`
- `long &iword (int __ix)`
- `char narrow (char_type __c, char __dfault) const`

- `void open` (const char \*\_\_s, `ios_base::openmode` \_\_mode=`ios_base::out|ios_base::trunc`)
- `void open` (const `std::string` &\_\_s, `ios_base::openmode` \_\_mode=`ios_base::out|ios_base::trunc`)
- `streamsize precision` () const
- `streamsize precision` (streamsize \_\_prec)
- `void *& pword` (int \_\_ix)
- `__filebuf_type * rdbuf` () const
- `basic_streambuf<_CharT, _Traits> * rdbuf` (`basic_streambuf<_CharT, _Traits> * __sb`)
- `iosstate rdstate` () const
- `void register_callback` (`event_callback` \_\_fn, int \_\_index)
- `__ostream_type & seekp` (`off_type`, `ios_base::seekdir`)
- `__ostream_type & seekp` (`pos_type`)
- `fmtflags setf` (`fmtflags` \_\_fmtfl)
- `fmtflags setf` (`fmtflags` \_\_fmtfl, `fmtflags` \_\_mask)
- `void setstate` (`iosstate` \_\_state)
- `pos_type tellp` ()
- `basic_ostream<_CharT, _Traits> * tie` () const
- `basic_ostream<_CharT, _Traits> * tie` (`basic_ostream<_CharT, _Traits> * __tistr`)
- `void unsetf` (`fmtflags` \_\_mask)
- `char_type widen` (char \_\_c) const
- `streamsize width` (streamsize \_\_wide)
- `streamsize width` () const
- `__ostream_type & operator<<` (`__ostream_type` &(\_\_pf)(`__ostream_type` &))
- `__ostream_type & operator<<` (`__ios_type` &(\_\_pf)(`__ios_type` &))
- `__ostream_type & operator<<` (`ios_base` &(\_\_pf)(`ios_base` &))

### Arithmetic Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ofstream::sentry`. This can have several effects, concluding with the setting of a status flag; see the `sentry` documentation for more.

If the `sentry` status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__ostream_type & operator<<` (long \_\_n)

- `__ostream_type & operator<< (unsigned long __n)`
- `__ostream_type & operator<< (bool __n)`
- `__ostream_type & operator<< (short __n)`
- `__ostream_type & operator<< (unsigned short __n)`
- `__ostream_type & operator<< (int __n)`
- `__ostream_type & operator<< (unsigned int __n)`
- `__ostream_type & operator<< (long long __n)`
- `__ostream_type & operator<< (unsigned long long __n)`
- `__ostream_type & operator<< (double __f)`
- `__ostream_type & operator<< (float __f)`
- `__ostream_type & operator<< (long double __f)`
- `__ostream_type & operator<< (const void *__p)`
- `__ostream_type & operator<< (__streambuf_type *__sb)`

### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If badbit is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type & put (char_type __c)`
- `void _M_write (const char_type *__s, streamsize __n)`
- `__ostream_type & write (const char_type *__s, streamsize __n)`
- `operator void * () const`
- `bool operator! () const`

### Static Public Member Functions

- static bool `sync_with_stdio` (bool \_\_sync=true)
- static int `xalloc` () throw ()

### Static Public Attributes

- static const `fmtflags adjustfield`
- static const `openmode app`
- static const `openmode ate`
- static const `iosstate badbit`
- static const `fmtflags basefield`

- static const [seekdir](#) `beg`
- static const [openmode](#) `binary`
- static const [fmtflags](#) `boolalpha`
- static const [seekdir](#) `cur`
- static const [fmtflags](#) `dec`
- static const [seekdir](#) `end`
- static const [iostate](#) `eofbit`
- static const [iostate](#) `failbit`
- static const [fmtflags](#) `fixed`
- static const [fmtflags](#) `floatfield`
- static const [iostate](#) `goodbit`
- static const [fmtflags](#) `hex`
- static const [openmode](#) `in`
- static const [fmtflags](#) `internal`
- static const [fmtflags](#) `left`
- static const [fmtflags](#) `oct`
- static const [openmode](#) `out`
- static const [fmtflags](#) `right`
- static const [fmtflags](#) `scientific`
- static const [fmtflags](#) `showbase`
- static const [fmtflags](#) `showpoint`
- static const [fmtflags](#) `showpos`
- static const [fmtflags](#) `skipws`
- static const [openmode](#) `trunc`
- static const [fmtflags](#) `unitbuf`
- static const [fmtflags](#) `uppercase`

### Protected Types

- enum { `_S_local_word_size` }

### Protected Member Functions

- void `_M_cache_locale` (const [locale](#) &\_\_loc)
- void `_M_call_callbacks` ([event](#) \_\_ev) throw ()
- void `_M_dispose_callbacks` (void) throw ()
- `_Words` & `_M_grow_words` (int \_\_index, bool \_\_iword)
- void `_M_init` () throw ()
- template<typename \_ValueT >  
    [\\_\\_ostream\\_type](#) & `_M_insert` (\_ValueT \_\_v)
- void `init` ([basic\\_streambuf](#)<\_CharT, \_Traits> \*\_\_sb)



### Protected Attributes

- `_Callback_list * _M_callbacks`
- `const __ctype_type * _M_ctype`
- `iostate _M_exception`
- `char_type _M_fill`
- `bool _M_fill_init`
- `fmtflags _M_flags`
- `locale _M_ios_locale`
- `_Words _M_local_word [_S_local_word_size]`
- `const __num_get_type * _M_num_get`
- `const __num_put_type * _M_num_put`
- `streamsize _M_precision`
- `basic_streambuf<_CharT, _Traits> * _M_streambuf`
- `iostate _M_streambuf_state`
- `basic_ostream<_CharT, _Traits> * _M_tie`
- `streamsize _M_width`
- `_Words * _M_word`
- `int _M_word_size`
- `_Words _M_word_zero`

### Friends

- class `sentry`

#### 5.384.1 Detailed Description

`template<typename _CharT, typename _Traits> class std::basic_ofstream< _CharT, _Traits>`

Controlling output for files.

This class supports reading from named files, using the inherited functions from `std::basic_ostream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.

Definition at line 588 of file `fstream`.

#### 5.384.2 Member Typedef Documentation

**5.384.2.1** `template<typename _CharT, typename _Traits> typedef  
    ctype<_CharT> std::basic_ostream<_CharT, _Traits>  
    ::__ctype_type [inherited]`

These are non-standard types.

Reimplemented from [std::basic\\_ios<\\_CharT, \\_Traits>](#).

Definition at line 73 of file ostream.

**5.384.2.2** `template<typename _CharT, typename _Traits> typedef  
num_get<_CharT, istreambuf_iterator<_CharT, _Traits>  
> std::basic_ios<_CharT, _Traits>::__num_get_type  
[inherited]`

These are non-standard types.

Reimplemented in [std::basic\\_istream<\\_CharT, \\_Traits>](#), [std::basic\\_istream<char, \\_Traits>](#), and [std::basic\\_istream<char>](#).

Definition at line 88 of file basic\_ios.h.

**5.384.2.3** `template<typename _CharT, typename _Traits> typedef  
num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>  
> std::basic_ostream<_CharT, _Traits>::__num_put_type  
[inherited]`

These are non-standard types.

Reimplemented from [std::basic\\_ios<\\_CharT, \\_Traits>](#).

Definition at line 72 of file ostream.

**5.384.2.4** `template<typename _CharT, typename _Traits> typedef _CharT  
std::basic_ofstream<_CharT, _Traits>::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_ostream<\\_CharT, \\_Traits>](#).

Definition at line 592 of file fstream.

**5.384.2.5** `typedef void(* std::ios_base::event_callback)(event, ios_base &, int)  
[inherited]`

The type of an event callback function.

#### Parameters

*event* One of the members of the event enum.

*ios\_base* Reference to the `ios_base` object.

*int* The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 438 of file `ios_base.h`.

#### 5.384.2.6 `typedef _Ios_Fmtflags std::ios_base::fmtflags` [inherited]

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 257 of file `ios_base.h`.

**5.384.2.7** `template<typename _CharT, typename _Traits> typedef  
traits_type::int_type std::basic_ofstream<_CharT, _Traits  
>::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_ostream<\\_CharT, \\_Traits>](#).

Definition at line 594 of file `fstream`.

**5.384.2.8** `typedef _Ios_Iostate std::ios_base::iostate [inherited]`

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 332 of file `ios_base.h`.

**5.384.2.9** `template<typename _CharT, typename _Traits> typedef  
traits_type::off_type std::basic_ofstream<_CharT, _Traits  
>::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_ostream<\\_CharT, \\_Traits>](#).

Definition at line 596 of file `fstream`.

**5.384.2.10** `typedef _Ios_Openmode std::ios_base::openmode [inherited]`

This is a bitmask type.

*\_Ios\_Openmode* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type *openmode* are:

- *app*
- *ate*
- *binary*
- *in*
- *out*
- *trunc*

Definition at line 363 of file *ios\_base.h*.

**5.384.2.11** `template<typename _CharT , typename _Traits > typedef traits_type::pos_type std::basic_ofstream< _CharT, _Traits >::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_ostream<\\_CharT, \\_Traits>](#).

Definition at line 595 of file *fstream*.

**5.384.2.12** `typedef _Ios_Seekdir std::ios_base::seekdir [inherited]`

This is an enumerated type.

*\_Ios\_Seekdir* is implementation-defined. Defined values of type *seekdir* are:

- *beg*
- *cur*, equivalent to *SEEK\_CUR* in the C standard library.
- *end*, equivalent to *SEEK\_END* in the C standard library.

Definition at line 395 of file *ios\_base.h*.

**5.384.2.13** `template<typename _CharT, typename _Traits> typedef _Traits  
std::basic_ofstream<_CharT, _Traits>::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_ostream<\\_CharT, \\_Traits>](#).

Definition at line 593 of file fstream.

**5.384.3 Member Enumeration Documentation**

**5.384.3.1** `enum std::ios_base::event [inherited]`

The set of events that may be passed to an event callback.

erase\_event is used during ~ios() and copyfmt(). imbue\_event is used during [imbue\(\)](#). copyfmt\_event is used during copyfmt().

Definition at line 421 of file ios\_base.h.

**5.384.4 Constructor & Destructor Documentation**

**5.384.4.1** `template<typename _CharT, typename _Traits>  
std::basic_ofstream<_CharT, _Traits>::basic_ofstream ( )  
[inline]`

Default constructor.

Initializes sb using its default constructor, and passes &sb to the base class initializer. Does not open any files (you haven't given it a filename to open).

Definition at line 614 of file fstream.

References [std::basic\\_ios<\\_CharT, \\_Traits>::init\(\)](#).

**5.384.4.2** `template<typename _CharT, typename _Traits>  
std::basic_ofstream<_CharT, _Traits>::basic_ofstream ( const char  
* __s, ios_base::openmode __mode = ios_base::out|ios_base::trunc )  
[inline, explicit]`

Create an output file stream.

**Parameters**

*s* Null terminated string specifying the filename.

*mode* Open file in specified mode (see [std::ios\\_base](#)).

[ios\\_base::out](#)|[ios\\_base::trunc](#) is automatically included in *mode*.

Tip: When using `std::string` to hold the filename, you must use `.c_str()` before passing it to this constructor.

Definition at line 629 of file `fstream`.

References `std::basic_ios<_CharT, _Traits>::init()`, and `std::basic_ofstream<_CharT, _Traits>::open()`.

```
5.384.4.3 template<typename _CharT , typename _Traits >
std::basic_ofstream< _CharT, _Traits >::basic_ofstream
(const std::string & __s, ios_base::openmode __mode =
ios_base::out|ios_base::trunc) [inline, explicit]
```

Create an output file stream.

**Parameters**

*s* `std::string` specifying the filename.

*mode* Open file in specified mode (see [std::ios\\_base](#)).

[ios\\_base::out](#)|[ios\\_base::trunc](#) is automatically included in *mode*.

Definition at line 647 of file `fstream`.

References `std::basic_ios<_CharT, _Traits>::init()`, and `std::basic_ofstream<_CharT, _Traits>::open()`.

```
5.384.4.4 template<typename _CharT , typename _Traits >
std::basic_ofstream< _CharT, _Traits >::~~basic_ofstream ()
[inline]
```

The destructor does nothing.

The file is closed by the `filebuf` object, not the formatting stream.

Definition at line 662 of file `fstream`.

### 5.384.5 Member Function Documentation

#### 5.384.5.1 const locale& std::ios\_base::\_M\_getloc ( ) const [inline, inherited]

Locale access.

##### Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 708 of file ios\_base.h.

Referenced by std::money\_get<\_CharT, \_InIter>::do\_get(), std::num\_get<\_CharT, \_InIter>::do\_get(), std::time\_get<\_CharT, \_InIter>::do\_get\_date(), std::time\_get<\_CharT, \_InIter>::do\_get\_monthname(), std::time\_get<\_CharT, \_InIter>::do\_get\_time(), std::time\_get<\_CharT, \_InIter>::do\_get\_weekday(), std::time\_get<\_CharT, \_InIter>::do\_get\_year(), std::time\_put<\_CharT, \_OutIter>::do\_put(), std::num\_put<\_CharT, \_OutIter>::do\_put(), and std::time\_put<\_CharT, \_OutIter>::put().

#### 5.384.5.2 template<typename \_CharT, typename \_Traits> void std::basic\_ostream<\_CharT, \_Traits>::\_M\_write ( const char\_type \* \_\_s, streamsize \_\_n ) [inline, inherited]

Simple insertion.

##### Parameters

*c* The character to insert.

##### Returns

\*this

Tries to insert *c*.

##### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 289 of file ostream.

Referenced by std::basic\_ostream<\_CharT, \_Traits>::write().



**5.384.5.3** `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::bad ( ) const [inline, inherited]`

Fast error checking.

#### Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 203 of file `basic_ios.h`.

**5.384.5.4** `template<typename _CharT, typename _Traits> void std::basic_ios<_CharT, _Traits>::clear ( iostate __state = goodbit ) [inherited]`

[Re]sets the error state.

#### Parameters

*state* The new state flag(s) to set.

See [std::ios\\_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

Definition at line 42 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::exceptions()`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::rdstate()`.

Referenced by `std::basic_ios<char, _Traits>::exceptions()`, `std::basic_fstream<_CharT, _Traits>::open()`, `std::basic_ofstream<_CharT, _Traits>::open()`, `std::basic_ifstream<_CharT, _Traits>::open()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ios<char, _Traits>::setstate()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

**5.384.5.5** `template<typename _CharT, typename _Traits> void std::basic_ofstream<_CharT, _Traits>::close ( ) [inline]`

Close the file.

## 5.384 std::basic\_ofstream<\_CharT, \_Traits> Class Template Reference 2023

---

Calls `std::basic_filebuf::close()`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 742 of file `fstream`.

References `std::basic_filebuf<_CharT, _Traits>::close()`, `std::ios_base::failbit`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.384.5.6** `template<typename _CharT, typename _Traits> basic_ios<_CharT, _Traits> & std::basic_ios<_CharT, _Traits>::copyfmt (const basic_ios<_CharT, _Traits> & __rhs) [inherited]`

Copies fields of `__rhs` into this.

### Parameters

`__rhs` The source values for the copies.

### Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy `exceptions()`.

Definition at line 64 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::exceptions()`, `std::basic_ios<_CharT, _Traits>::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios<_CharT, _Traits>::tie()`, and `std::ios_base::width()`.

**5.384.5.7** `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::eof() const [inline, inherited]`

Fast error checking.

### Returns

True if the `eofbit` is set.

Note that other `iostate` flags may also be set.

Definition at line 182 of file `basic_ios.h`.

## 5.384 std::basic\_ofstream<\_CharT, \_Traits> Class Template Reference 2024

---

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

**5.384.5.8** `template<typename _CharT, typename _Traits> iostate  
std::basic_ios<_CharT, _Traits>::exceptions ( ) const [inline,  
inherited]`

Throwing exceptions on errors.

### Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of [exceptions\(iostate\)](#) for the meaning of the return value.

Definition at line 214 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::clear()`, and `std::basic_ios<_CharT, _Traits>::copyfmt()`.

**5.384.5.9** `template<typename _CharT, typename _Traits> void std::basic_ios<  
_CharT, _Traits>::exceptions ( iostate __except ) [inline,  
inherited]`

Throwing exceptions on errors.

### Parameters

*except* The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type [std::ios\\_base::failure](#) is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>
```

```
int main()
{
 std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

 std::ifstream f ("/etc/motd");

 std::cerr << "Setting badbit\n";
 f.setstate (std::ios_base::badbit);

 std::cerr << "Setting exception mask\n";
 f.exceptions (std::ios_base::badbit);
}
```

Definition at line 249 of file basic\_ios.h.

**5.384.5.10** `template<typename _CharT, typename _Traits> bool  
std::basic_ios<_CharT, _Traits>::fail ( ) const [inline,  
inherited]`

Fast error checking.

#### Returns

True if either the badbit or the failbit is set.

Checking the badbit in [fail\(\)](#) is historical practice. Note that other iostate flags may also be set.

Definition at line 193 of file basic\_ios.h.

Referenced by `std::basic_ios<char, _Traits>::operator void *()`, `std::basic_ios<char, _Traits>::operator!()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ofstream<_CharT, _Traits>::seekp()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_ofstream<_CharT, _Traits>::tellp()`, and `std::regex_traits<_Ch_type>::value()`.

**5.384.5.11** `template<typename _CharT, typename _Traits> char_type  
std::basic_ios<_CharT, _Traits>::fill ( char_type __ch )  
[inline, inherited]`

Sets a new *empty* character.

#### Parameters

*ch* The new character.

### Returns

The previous fill character.

The fill character is used to fill out space when  $P+$  characters have been requested (e.g., via `setw`),  $Q$  characters are actually used, and  $Q < P$ . It defaults to a space ( ' ') in the current locale.

Definition at line 382 of file `basic_ios.h`.

**5.384.5.12** `template<typename _CharT, typename _Traits> char_type  
std::basic_ios<_CharT, _Traits>::fill ( ) const [inline,  
inherited]`

Retrieves the *empty* character.

### Returns

The current fill character.

It defaults to a space ( ' ') in the current locale.

Definition at line 362 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, and `std::basic_ios<char, _Traits>::fill()`.

**5.384.5.13** `fmtflags std::ios_base::flags ( ) const [inline, inherited]`

Access to format flags.

### Returns

The format control flags for both input and output.

Definition at line 553 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, `std::basic_ofstream<_CharT, _Traits>::operator<<()`, `std::operator<<()`, `std::operator>>()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

**5.384.5.14** `fmtflags std::ios_base::flags ( fmtflags __fmtfl ) [inline, inherited]`

Setting new format flags all at once.

**Parameters**

*fmtfl* The new flags to set.

**Returns**

The previous format control flags.

This function overwrites all the format flags with *fmtfl*.

Definition at line 564 of file `ios_base.h`.

**5.384.5.15** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::flush ( ) [inherited]`

Synchronizing the stream buffer.

**Returns**

\*this

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf() ->pubsync()`, and if that returns -1, sets `badbit`.

Definition at line 213 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

Referenced by `std::flush()`.

**5.384.5.16** `locale std::ios_base::getloc ( ) const [inline, inherited]`

Locale access.

**Returns**

A copy of the current locale.

## 5.384 `std::basic_ofstream<_CharT, _Traits>` Class Template Reference 2028

---

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 697 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::money_put<_CharT, _Outputter>::do_put()`, `std::basic_ios<_CharT, _Traits>::imbue()`, `std::operator>>()`, and `std::ws()`.

**5.384.5.17** `template<typename _CharT, typename _Traits> bool  
std::basic_ios<_CharT, _Traits>::good( ) const [inline,  
inherited]`

Fast error checking.

### Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 172 of file `basic_ios.h`.

Referenced by `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

**5.384.5.18** `template<typename _CharT, typename _Traits> locale  
std::basic_ios<_CharT, _Traits>::imbue( const locale & __loc )  
[inherited]`

Moves to a new locale.

### Parameters

*loc* The new locale.

### Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Reimplemented from [std::ios\\_base](#).

## 5.384 `std::basic_ofstream<_CharT, _Traits>` Class Template Reference 2029

Definition at line 115 of file `basic_ios.tcc`.

References `std::ios_base::getloc()`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

Referenced by `std::operator<<()`.

**5.384.5.19** `template<typename _CharT, typename _Traits> void  
std::basic_ios<_CharT, _Traits>::init ( basic_streambuf<  
_CharT, _Traits> * __sb ) [protected, inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 127 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

Referenced by `std::basic_fstream<_CharT, _Traits>::basic_fstream()`, `std::basic_ifstream<_CharT, _Traits>::basic_ifstream()`, `std::basic_ios<char, _Traits>::basic_ios()`, `std::basic_istream<char>::basic_istream()`, `std::basic_istringstream<_CharT, _Traits, _Alloc>::basic_istringstream()`, `std::basic_ofstream<_CharT, _Traits>::basic_ofstream()`, `std::basic_ostream<char>::basic_ostream()`, `std::basic_ostringstream<_CharT, _Traits, _Alloc>::basic_ostringstream()`, and `std::basic_stringstream<_CharT, _Traits, _Alloc>::basic_stringstream()`.

**5.384.5.20** `template<typename _CharT, typename _Traits> bool  
std::basic_ofstream<_CharT, _Traits>::is_open ( ) [inline]`

Wrapper to test for an open file.

### Returns

`rdbuf()` -> `is_open()`

Definition at line 681 of file `fstream`.

References `std::basic_filebuf<_CharT, _Traits>::is_open()`.

**5.384.5.21** `long& std::ios_base::iword ( int __ix ) [inline, inherited]`

Access to integer array.



**Parameters**

`__ix` Index into the array.

**Returns**

A reference to an integer associated with the index.

The `ixword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 743 of file `ios_base.h`.

**5.384.5.22** `template<typename _CharT, typename _Traits> char  
std::basic_ios<_CharT, _Traits>::narrow( char_type __c, char  
__dfault ) const [inline, inherited]`

Squeezes characters.

**Parameters**

`c` The character to narrow.

`dfault` The character to narrow.

**Returns**

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, dfault)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.ht>

Definition at line 422 of file `basic_ios.h`.

**5.384.5.23** `template<typename _CharT, typename _Traits> void  
std::basic_ofstream<_CharT, _Traits>::open( const char * __s,  
ios_base::openmode __mode = ios_base::out | ios_base::trunc )  
[inline]`

Opens an external file.

#### Parameters

- s* The name of the file.
- mode* The open mode flags.

Calls `std::basic_filebuf::open(s,mode|out|trunc)`. If that function fails, `failbit` is set in the stream's error state.

Tip: When using `std::string` to hold the filename, you must use `.c_str()` before passing it to this constructor.

Definition at line 702 of file `fstream`.

References `std::basic_ios<_CharT, _Traits>::clear()`, `std::ios_base::failbit`, `std::basic_filebuf<_CharT, _Traits>::open()`, `std::ios_base::out`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

Referenced by `std::basic_ofstream<_CharT, _Traits>::basic_ofstream()`.

```
5.384.5.24 template<typename _CharT, typename _Traits> void
std::basic_ofstream<_CharT, _Traits>::open (const std::string &
__s, ios_base::openmode __mode = ios_base::out | ios_base::trunc
) [inline]
```

Opens an external file.

#### Parameters

- s* The name of the file.
- mode* The open mode flags.

Calls `std::basic_filebuf::open(s,mode|out|trunc)`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 723 of file `fstream`.

References `std::basic_ios<_CharT, _Traits>::clear()`, `std::ios_base::failbit`, `std::basic_filebuf<_CharT, _Traits>::open()`, `std::ios_base::out`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

```
5.384.5.25 template<typename _CharT, typename _Traits> std::basic_ios<
_CharT, _Traits>::operator void * () const [inline,
inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 113 of file `basic_ios.h`.

```
5.384.5.26 template<typename _CharT, typename _Traits> bool
std::basic_ios<_CharT, _Traits>::operator! () const
[inline, inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 117 of file `basic_ios.h`.

```
5.384.5.27 template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream<_CharT, _Traits>::operator<< (__ios_type
&(*)(__ios_type &) __pf) [inline, inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

Definition at line 119 of file `ostream`.

```
5.384.5.28 template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream<_CharT, _Traits>::operator<< (unsigned int
__n) [inline, inherited]
```

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 193 of file `ostream`.

**5.384.5.29** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<<( __streambuf_type * __sb ) [inherited]`

Extracting from another streambuf.

#### Parameters

*sb* A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from *sb* and inserted into *\*this* until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from *sb*, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 122 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.384.5.30** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( ios_base &(__pf) ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the `iomanip` header.

Definition at line 129 of file `ostream`.

**5.384.5.31** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ofstream<_CharT, _Traits>::operator<< ( unsigned  
long long __n ) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 206 of file `ostream`.

**5.384.5.32** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ofstream<_CharT, _Traits>::operator<< ( double __f  
) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 211 of file `ostream`.

**5.384.5.33** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ofstream<_CharT, _Traits>::operator<<( float __f )  
[inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 215 of file `ostream`.

**5.384.5.34** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ofstream<_CharT, _Traits>::operator<<( long __n )  
[inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 167 of file `ostream`.

**5.384.5.35** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ofstream<_CharT, _Traits>::operator<<( long double  
__f ) [inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 223 of file ostream.

**5.384.5.36** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ofstream<_CharT, _Traits>::operator<<( const void * __p ) [inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 227 of file ostream.

**5.384.5.37** `template<typename _CharT, typename _Traits> basic_ofstream<_CharT, _Traits> & std::basic_ofstream<_CharT, _Traits>::operator<<( int __n ) [inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 108 of file `ostream.tcc`.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

**5.384.5.38** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ofstream<_CharT, _Traits>::operator<<( unsigned  
long __n ) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 171 of file `ostream`.

**5.384.5.39** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ofstream<_CharT, _Traits>::operator<<( bool __n )  
[inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 175 of file `ostream`.



**5.384.5.40** `template<typename _CharT, typename _Traits> basic_ofstream<_CharT, _Traits> & std::basic_ofstream<_CharT, _Traits>::operator<<( short __n ) [inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 94 of file `ostream.tcc`.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

**5.384.5.41** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ofstream<_CharT, _Traits>::operator<<( long long __n ) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 202 of file `ostream`.

**5.384.5.42** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ofstream<_CharT, _Traits>::operator<<( unsigned short __n ) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 182 of file `ostream`.

**5.384.5.43** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ofstream<_CharT, _Traits>::operator<< ( __ostream_type &(*)(__ostream_type &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

Definition at line 110 of file `ostream`.

**5.384.5.44** `streamsize std::ios_base::precision ( ) const [inline, inherited]`

Flags access.

#### Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 623 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, and `std::operator<<()`.

**5.384.5.45** `streamsize std::ios_base::precision ( streamsize __prec ) [inline, inherited]`

Changing flags.

**Parameters**

*prec* The new precision value.

**Returns**

The previous value of `precision()`.

Definition at line 632 of file `ios_base.h`.

**5.384.5.46** `template<typename _CharT, typename _Traits> basic_ofstream<  
_CharT, _Traits> & std::basic_ofstream<_CharT, _Traits>::put (  
char_type __c ) [inherited]`

Simple insertion.

**Parameters**

*c* The character to insert.

**Returns**

\*this

Tries to insert *c*.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 151 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

Referenced by `std::endl()`, and `std::ends()`.

**5.384.5.47** `void*& std::ios_base::pword ( int __ix ) [inline,  
inherited]`

Access to void pointer array.

**Parameters**

`__ix` Index into the array.

**Returns**

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 764 of file `ios_base.h`.

```
5.384.5.48 template<typename _CharT, typename _Traits> __filebuf_type*
 std::basic_ofstream<_CharT, _Traits>::rdbuf () const
 [inline]
```

Accessing the underlying buffer.

**Returns**

The current `basic_filebuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Definition at line 673 of file `fstream`.

```
5.384.5.49 template<typename _CharT, typename _Traits> basic_streambuf<
 _CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rdbuf (
 basic_streambuf<_CharT, _Traits> * __sb) [inherited]
```

Changing the underlying buffer.

**Parameters**

`sb` The new stream buffer.

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream foo; // or some other derived type
std::streambuf* p =;

foo.ios::rdbuf(p); // ios == basic_ios<char>
```

Definition at line 54 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::clear()`.

**5.384.5.50** `template<typename _CharT, typename _Traits> iostate  
std::basic_ios<_CharT, _Traits>::rdstate ( ) const [inline,  
inherited]`

Returns the error state of the stream buffer.

#### Returns

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 129 of file `basic_ios.h`.

Referenced by `std::basic_ios<char, _Traits>::bad()`, `std::basic_ios<_CharT, _Traits>::clear()`, `std::basic_ios<char, _Traits>::eof()`, `std::basic_ios<char, _Traits>::fail()`, `std::basic_ios<char, _Traits>::good()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ios<char, _Traits>::setstate()`, and `std::basic_istream<_CharT, _Traits>::unset()`.

**5.384.5.51** `void std::ios_base::register_callback ( event_callback __fn, int  
__index ) [inherited]`

Add the callback `__fn` with parameter `__index`.

#### Parameters

`__fn` The function to add.

*\_\_index* The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**5.384.5.52** `template<typename _CharT, typename _Traits> basic_ofstream<_CharT, _Traits> & std::basic_ofstream<_CharT, _Traits>::seekp( off_type __off, ios_base::seekdir __dir ) [inherited]`

Changing the current write position.

#### Parameters

*off* A file offset object.

*dir* The direction in which to seek.

#### Returns

\*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

Definition at line 292 of file ostream.tcc.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.384.5.53** `template<typename _CharT, typename _Traits> basic_ofstream<_CharT, _Traits> & std::basic_ofstream<_CharT, _Traits>::seekp( pos_type __pos ) [inherited]`

Changing the current write position.

#### Parameters

*pos* A file position object.

#### Returns

\*this

## 5.384 std::basic\_ofstream<\_CharT, \_Traits> Class Template Reference 2044

---

If `fail()` is not true, calls `rdbuf() ->pubseekpos(pos)`. If that function fails, sets failbit.

Definition at line 260 of file ostream.tcc.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

### 5.384.5.54 `fmtflags std::ios_base::setf( fmtflags __fmtfl, fmtflags __mask )` [inline, inherited]

Setting new format flags.

#### Parameters

*fmtfl* Additional flags to set.

*mask* The flags mask for *fmtfl*.

#### Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 597 of file ios\_base.h.

### 5.384.5.55 `fmtflags std::ios_base::setf( fmtflags __fmtfl )` [inline, inherited]

Setting new format flags.

#### Parameters

*fmtfl* Additional flags to set.

#### Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 580 of file ios\_base.h.

Referenced by std::dec(), std::fixed(), std::hex(), std::left(), std::oct(), std::right(), std::scientific(), std::showbase(), std::showpoint(), std::showpos(), std::skipws(), std::unitbuf(), and std::uppercase().

**5.384.5.56** `template<typename _CharT, typename _Traits> void  
std::basic_ios<_CharT, _Traits>::setstate ( iostate __state )  
[inline, inherited]`

Sets additional flags in the error state.

#### Parameters

*state* The additional state flag(s) to set.

See [std::ios\\_base::iostate](#) for the possible bit values.

Definition at line 149 of file basic\_ios.h.

Referenced by std::basic\_ostream<char>::\_M\_write(), std::basic\_fstream<\_CharT, \_Traits>::close(), std::basic\_ofstream<\_CharT, \_Traits>::close(), std::basic\_ifstream<\_CharT, \_Traits>::close(), std::basic\_ostream<\_CharT, \_Traits>::flush(), std::basic\_istream<\_CharT, \_Traits>::get(), std::basic\_istream<\_CharT, \_Traits>::getline(), std::getline(), std::basic\_istream<\_CharT, \_Traits>::ignore(), std::basic\_fstream<\_CharT, \_Traits>::open(), std::basic\_ofstream<\_CharT, \_Traits>::open(), std::basic\_ifstream<\_CharT, \_Traits>::open(), std::basic\_ostream<\_CharT, \_Traits>::operator<<(), std::basic\_istream<\_CharT, \_Traits>::operator>>(), std::operator>>(), std::basic\_istream<\_CharT, \_Traits>::peek(), std::basic\_ostream<\_CharT, \_Traits>::put(), std::basic\_istream<\_CharT, \_Traits>::putback(), std::basic\_istream<\_CharT, \_Traits>::read(), std::basic\_istream<\_CharT, \_Traits>::readsome(), std::basic\_istream<\_CharT, \_Traits>::seekg(), std::basic\_ostream<\_CharT, \_Traits>::seekp(), std::basic\_ostream<\_CharT, \_Traits>::sentry::sentry(), std::basic\_istream<\_CharT, \_Traits>::sentry::sentry(), std::basic\_istream<\_CharT, \_Traits>::sync(), std::basic\_istream<\_CharT, \_Traits>::unget(), and std::ws().

**5.384.5.57** `static bool std::ios_base::sync_with_stdio ( bool __sync = true )  
[static, inherited]`

Interaction with the standard C I/O objects.



**Parameters**

*sync* Whether to synchronize or not.

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

**5.384.5.58** `template<typename _CharT, typename _Traits> basic_ofstream<_CharT, _Traits>::pos_type std::basic_ofstream<_CharT, _Traits>::tellp ( ) [inherited]`

Getting the current write position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, out)`.

Definition at line 239 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::out`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

**5.384.5.59** `template<typename _CharT, typename _Traits> basic_ofstream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie ( basic_ofstream<_CharT, _Traits> * __tiestr ) [inline, inherited]`

Ties this stream to an output stream.

**Parameters**

*tiestr* The output stream.

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see [tie\(\)](#) for more.

Definition at line 299 of file basic\_ios.h.

```
5.384.5.60 template<typename _CharT, typename _Traits>
 basic_ofstream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie() const [inline, inherited]
```

Fetches the current *tied* stream.

#### Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, [std::cin](#) is tied to [std::cout](#).

Definition at line 287 of file basic\_ios.h.

Referenced by [std::basic\\_ios<\\_CharT, \\_Traits>::copyfmt\(\)](#), [std::basic\\_ofstream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#), and [std::basic\\_istream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#).

```
5.384.5.61 void std::ios_base::unsetf (fmtflags __mask) [inline,
 inherited]
```

Clearing format flags.

#### Parameters

*mask* The flags to unset.

This function clears *mask* in the format flags.

Definition at line 612 of file ios\_base.h.

Referenced by [std::noboolalpha\(\)](#), [std::noshowbase\(\)](#), [std::noshowpoint\(\)](#), [std::noshowpos\(\)](#), [std::noskipws\(\)](#), [std::nounitbuf\(\)](#), and [std::nouppercase\(\)](#).

```
5.384.5.62 template<typename _CharT, typename _Traits> char_type
 std::basic_ios<_CharT, _Traits>::widen (char __c) const
 [inline, inherited]
```

Widens characters.

**Parameters**

*c* The character to widen.

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 441 of file `basic_ios.h`.

Referenced by `std::endl()`, `std::basic_ios< char, _Traits >::fill()`, `std::basic_istream< char >::get()`, `std::basic_istream< char >::getline()`, `std::getline()`, and `std::operator>>()`.

**5.384.5.63 streamsize std::ios\_base::width ( streamsize *\_\_wide* ) [inline, inherited]**

Changing flags.

**Parameters**

*wide* The new width value.

**Returns**

The previous value of `width()`.

Definition at line 655 of file `ios_base.h`.

**5.384.5.64 streamsize std::ios\_base::width ( ) const [inline, inherited]**

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 646 of file ios\_base.h.

Referenced by std::basic\_ios<\_CharT, \_Traits>::copyfmt(), std::num\_put<\_CharT, \_OutIter>::do\_put(), and std::operator>>().

**5.384.5.65** `template<typename _CharT, typename _Traits> basic_ofstream<  
_CharT, _Traits> & std::basic_ofstream<_CharT, _Traits>::write  
( const char_type * __s, streamsize __n ) [inherited]`

Character string insertion.

#### Parameters

- s* The array to insert.
- n* Maximum number of characters to insert.

#### Returns

\*this

Characters are copied from *s* and inserted into the stream until one of the following happens:

- *n* characters are inserted
- inserting into the output sequence fails (in this case, badbit will be set in the stream's error state)

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 185 of file ostream.tcc.

References std::basic\_ofstream<\_CharT, \_Traits>::\_M\_write(), and std::ios\_base::badbit.

**5.384.5.66** `static int std::ios_base::xalloc ( ) throw () [static,  
inherited]`

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iwor`d and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iwor`d and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iwor`d and `pword` arrays.

**5.384.6 Member Data Documentation****5.384.6.1 const fmtflags std::ios\_base::adjustfield [static, inherited]**

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Definition at line 312 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

**5.384.6.2 const openmode std::ios\_base::app [static, inherited]**

Seek to end before each write.

Definition at line 366 of file `ios_base.h`.

**5.384.6.3 const openmode std::ios\_base::ate [static, inherited]**

Open and seek to end immediately after opening.

Definition at line 369 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::open()`.

**5.384.6.4 const iostate std::ios\_base::badbit [static, inherited]**

## 5.384 std::basic\_ofstream<\_CharT, \_Traits> Class Template Reference 2051

Indicates a loss of integrity in an input or output sequence (such /// as an irrecoverable read error from a file).

Definition at line 336 of file ios\_base.h.

Referenced by std::basic\_ostream< char >::\_M\_write(), std::basic\_ios< char, \_Traits >::bad(), std::basic\_ios< \_CharT, \_Traits >::clear(), std::basic\_ios< char, \_Traits >::fail(), std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_ios< \_CharT, \_Traits >::init(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::operator<<(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::sync(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_ostream< \_CharT, \_Traits >::tellp(), std::basic\_istream< \_CharT, \_Traits >::unget(), std::basic\_ostream< \_CharT, \_Traits >::write(), and std::basic\_ostream< \_CharT, \_Traits >::sentry::~sentry().

### 5.384.6.5 const fmtflags std::ios\_base::basefield [static, inherited]

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Definition at line 315 of file ios\_base.h.

Referenced by std::dec(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::hex(), std::oct(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

### 5.384.6.6 const seekdir std::ios\_base::beg [static, inherited]

Request a seek relative to the beginning of the stream.

Definition at line 398 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::seekpos().

### 5.384.6.7 const openmode std::ios\_base::binary [static, inherited]

Perform input and output in binary mode (as opposed to text mode). /// This is probably not what you think it is; see ///

## 5.384 std::basic\_ofstream<\_CharT, \_Traits> Class Template Reference 2052

<http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 374 of file ios\_base.h.

Referenced by std::basic\_filebuf<\_CharT, \_Traits>::showmanyc().

### 5.384.6.8 const fmtflags std::ios\_base::boolalpha [static, inherited]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 260 of file ios\_base.h.

Referenced by std::boolalpha(), std::num\_get<\_CharT, \_InIter>::do\_get(), std::num\_put<\_CharT, \_OutIter>::do\_put(), and std::noboolalpha().

### 5.384.6.9 const seekdir std::ios\_base::cur [static, inherited]

Request a seek relative to the current position within the sequence.

Definition at line 401 of file ios\_base.h.

Referenced by std::basic\_filebuf<\_CharT, \_Traits>::imbue(), std::basic\_filebuf<\_CharT, \_Traits>::overflow(), std::basic\_filebuf<\_CharT, \_Traits>::pbackfail(), std::basic\_stringbuf<\_CharT, \_Traits, \_Alloc>::seekoff(), std::basic\_filebuf<\_CharT, \_Traits>::seekoff(), std::basic\_istream<\_CharT, \_Traits>::tellg(), and std::basic\_ostream<\_CharT, \_Traits>::tellp().

### 5.384.6.10 const fmtflags std::ios\_base::dec [static, inherited]

Converts integer input or generates integer output in decimal base.

Definition at line 263 of file ios\_base.h.

Referenced by std::dec().

### 5.384.6.11 const seekdir std::ios\_base::end [static, inherited]

Request a seek relative to the current end of the sequence.

Definition at line 404 of file ios\_base.h.

Referenced by `std::basic_filebuf<_CharT, _Traits>::open()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`.

#### 5.384.6.12 `const iostate std::ios_base::eofbit` [**static, inherited**]

Indicates that an input operation reached the end of an input sequence.

Definition at line 339 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_date()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_time()`, `std::time_get<_CharT, _InIter>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::basic_ios<char, _Traits>::eof()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::readsome()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, `std::basic_istream<_CharT, _Traits>::unget()`, and `std::ws()`.

#### 5.384.6.13 `const iostate std::ios_base::failbit` [**static, inherited**]

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 344 of file `ios_base.h`.

Referenced by `std::basic_fstream<_CharT, _Traits>::close()`, `std::basic_ofstream<_CharT, _Traits>::close()`, `std::basic_ifstream<_CharT, _Traits>::close()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_date()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_time()`, `std::time_get<_CharT, _InIter>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::basic_ios<char, _Traits>::fail()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_fstream<_CharT, _Traits>::open()`, `std::basic_ofstream<_CharT, _Traits>::open()`, `std::basic_ifstream<_CharT, _Traits>::open()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.



**5.384.6.14** `const fmtflags std::ios_base::fixed` [`static`, `inherited`]

Generate floating-point output in fixed-point notation.

Definition at line 266 of file `ios_base.h`.

Referenced by `std::fixed()`.

**5.384.6.15** `const fmtflags std::ios_base::floatfield` [`static`, `inherited`]

A mask of `scientific|fixed`. Useful for the 2-arg form of `setf`.

Definition at line 318 of file `ios_base.h`.

Referenced by `std::fixed()`, and `std::scientific()`.

**5.384.6.16** `const iostate std::ios_base::goodbit` [`static`, `inherited`]

Indicates all is well.

Definition at line 347 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::basic_ofstream<_CharT, _Traits>::flush()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_ios<_CharT, _Traits>::init()`, `std::basic_ofstream<_CharT, _Traits>::operator<<()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_ofstream<_CharT, _Traits>::put()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::readsome()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ofstream<_CharT, _Traits>::seekp()`, `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, `std::basic_istream<_CharT, _Traits>::sync()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

**5.384.6.17** `const fmtflags std::ios_base::hex` [`static`, `inherited`]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 269 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), std::hex(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

#### **5.384.6.18 const openmode std::ios\_base::in [static, inherited]**

Open for input. Default for `ifstream` and `fstream`.

Definition at line 377 of file ios\_base.h.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::\_M\_set\_buffer(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_filebuf< \_CharT, \_Traits >::pbackfail(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::showmanyc(), std::basic\_filebuf< \_CharT, \_Traits >::showmanyc(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::underflow(), std::basic\_filebuf< \_CharT, \_Traits >::underflow(), and std::basic\_filebuf< \_CharT, \_Traits >::xsgetn().

#### **5.384.6.19 const fmtflags std::ios\_base::internal [static, inherited]**

Adds fill characters at a designated internal point in certain `///` generated output, or identical to `right` if no such point is `///` designated.

Definition at line 274 of file ios\_base.h.

Referenced by std::internal().

#### **5.384.6.20 const fmtflags std::ios\_base::left [static, inherited]**

Adds fill characters on the right (final positions) of certain `///` generated output. (I.e., the thing you print is flush left.).

Definition at line 278 of file ios\_base.h.

Referenced by std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::left().

**5.384.6.21 const fmtflags std::ios\_base::oct [static, inherited]**

Converts integer input or generates integer output in octal base.

Definition at line 281 of file ios\_base.h.

Referenced by std::oct(), and std::basic\_ostream<\_CharT, \_Traits>::operator<<().

**5.384.6.22 const openmode std::ios\_base::out [static, inherited]**

Open for output. Default for ofstream and fstream.

Definition at line 380 of file ios\_base.h.

Referenced by std::basic\_filebuf<char\_type, traits\_type>::\_M\_set\_buffer(), std::basic\_ofstream<\_CharT, \_Traits>::open(), std::basic\_stringbuf<\_CharT, \_Traits, \_Alloc>::overflow(), std::basic\_filebuf<\_CharT, \_Traits>::overflow(), std::basic\_stringbuf<\_CharT, \_Traits, \_Alloc>::pbackfail(), std::basic\_stringbuf<\_CharT, \_Traits, \_Alloc>::seekoff(), std::basic\_ostream<\_CharT, \_Traits>::seekp(), std::basic\_stringbuf<\_CharT, \_Traits, \_Alloc>::seekpos(), std::basic\_ostream<\_CharT, \_Traits>::tellp(), and std::basic\_filebuf<\_CharT, \_Traits>::xsputn().

**5.384.6.23 const fmtflags std::ios\_base::right [static, inherited]**

Adds fill characters on the left (initial positions) of certain /// generated output. (I.e., the thing you print is flush right.).

Definition at line 285 of file ios\_base.h.

Referenced by std::right().

**5.384.6.24 const fmtflags std::ios\_base::scientific [static, inherited]**

Generates floating-point output in scientific notation.

Definition at line 288 of file ios\_base.h.

Referenced by std::scientific().

**5.384.6.25 const fmtflags std::ios\_base::showbase [static, inherited]**

Generates a prefix indicating the numeric base of generated integer /// output.

Definition at line 292 of file ios\_base.h.

Referenced by std::noshowbase(), and std::showbase().

**5.384.6.26 const fmtflags std::ios\_base::showpoint [static, inherited]**

Generates a decimal-point character unconditionally in generated /// floating-point output.

Definition at line 296 of file ios\_base.h.

Referenced by std::noshowpoint(), and std::showpoint().

**5.384.6.27 const fmtflags std::ios\_base::showpos [static, inherited]**

Generates a + sign in non-negative generated numeric output.

Definition at line 299 of file ios\_base.h.

Referenced by std::noshowpos(), and std::showpos().

**5.384.6.28 const fmtflags std::ios\_base::skipws [static, inherited]**

Skips leading white space before certain input operations.

Definition at line 302 of file ios\_base.h.

Referenced by std::noskipws(), std::basic\_istream<\_CharT, \_Traits>::sentry::sentry(), and std::skipws().

**5.384.6.29 const openmode std::ios\_base::trunc [static, inherited]**

Open for input. Default for ofstream.

Definition at line 383 of file ios\_base.h.

**5.384.6.30** `const fmtflags std::ios_base::unitbuf` [`static`, `inherited`]

Flushes output after each output operation.

Definition at line 305 of file `ios_base.h`.

Referenced by `std::noinitbuf()`, `std::unitbuf()`, and `std::basic_ostream< _CharT, _Traits >::sentry::~sentry()`.

**5.384.6.31** `const fmtflags std::ios_base::uppercase` [`static`, `inherited`]

Replaces certain lowercase letters with their uppercase equivalents /// in generated output.

Definition at line 309 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, `std::nouppercase()`, and `std::uppercase()`.

The documentation for this class was generated from the following file:

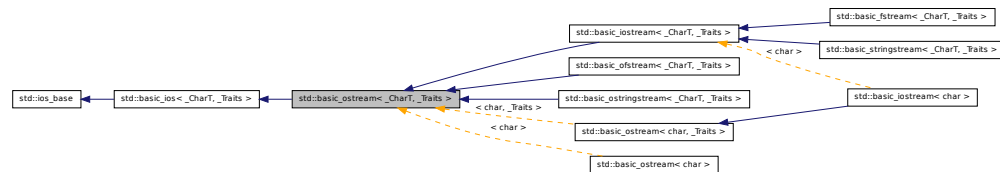
- [fstream](#)

**5.385** `std::basic_ostream< _CharT, _Traits >` Class Template Reference

Controlling output.

This is the base class for all output streams. It provides text formatting of all builtin types, and communicates with any class derived from [basic\\_streambuf](#) to do the actual output.

Inheritance diagram for `std::basic_ostream< _CharT, _Traits >`:



## Classes

- class [sentry](#)  
*Performs setup work for output streams.*

## Public Types

- typedef [ctype](#)<\_CharT> [\\_\\_ctype\\_type](#)
- typedef [basic\\_ios](#)<\_CharT, \_Traits> [\\_\\_ios\\_type](#)
- typedef [num\\_put](#)<\_CharT, [ostreambuf\\_iterator](#)<\_CharT, \_Traits>> [\\_\\_num\\_put\\_type](#)
- typedef [basic\\_ostream](#)<\_CharT, \_Traits> [\\_\\_ostream\\_type](#)
- typedef [basic\\_streambuf](#)<\_CharT, \_Traits> [\\_\\_streambuf\\_type](#)
- typedef \_CharT [char\\_type](#)
- enum [event](#) { [erase\\_event](#), [imbue\\_event](#), [copyfmt\\_event](#) }
- typedef void(\* [event\\_callback](#))([event](#), [ios\\_base](#) &, int)
- typedef \_Ios\_Fmtflags [fmtflags](#)
- typedef \_Traits::int\_type [int\\_type](#)
- typedef int [io\\_state](#)
- typedef \_Ios\_Iostate [iostate](#)
- typedef \_Traits::off\_type [off\\_type](#)
- typedef int [open\\_mode](#)
- typedef \_Ios\_Openmode [openmode](#)
- typedef \_Traits::pos\_type [pos\\_type](#)
- typedef int [seek\\_dir](#)
- typedef \_Ios\_Seekdir [seekdir](#)
- typedef [std::streamoff](#) [streamoff](#)
- typedef [std::streampos](#) [streampos](#)
- typedef \_Traits [traits\\_type](#)
  
- typedef [num\\_get](#)<\_CharT, [istreambuf\\_iterator](#)<\_CharT, \_Traits>> [\\_\\_num\\_get\\_type](#)

## Public Member Functions

- [basic\\_ostream](#) ([\\_\\_streambuf\\_type](#) \*\_\_sb)
- virtual [~basic\\_ostream](#) ()
- const [locale](#) & [\\_M\\_getloc](#) () const
- void [\\_M\\_setstate](#) ([iostate](#) \_\_state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) \_\_state=[goodbit](#))

- `basic_ios` & `copyfmt` (const `basic_ios` &\_\_rhs)
  - `bool eof` () const
  - `void exceptions` (iostate \_\_except)
  - `iostate exceptions` () const
  - `bool fail` () const
  - `char_type fill` () const
  - `char_type fill` (char\_type \_\_ch)
  - `fmtflags flags` () const
  - `fmtflags flags` (fmtflags \_\_fmtfl)
  - `__ostream_type` & `flush` ()
  - `locale getloc` () const
  - `bool good` () const
  - `locale imbue` (const `locale` &\_\_loc)
  - `long & iword` (int \_\_ix)
  - `char narrow` (char\_type \_\_c, char \_\_dfault) const
  - `streamsize precision` () const
  - `streamsize precision` (streamsize \_\_prec)
  - `void *& pword` (int \_\_ix)
  - `basic_streambuf<_CharT, _Traits> * rdbuf` () const
  - `basic_streambuf<_CharT, _Traits> * rdbuf` (`basic_streambuf<_CharT, _Traits> * __sb`)
  - `iostate rdstate` () const
  - `void register_callback` (event\_callback \_\_fn, int \_\_index)
  - `__ostream_type` & `seekp` (pos\_type)
  - `__ostream_type` & `seekp` (off\_type, ios\_base::seekdir)
  - `fmtflags setf` (fmtflags \_\_fmtfl)
  - `fmtflags setf` (fmtflags \_\_fmtfl, fmtflags \_\_mask)
  - `void setstate` (iostate \_\_state)
  - `pos_type tellp` ()
  - `basic_ostream<_CharT, _Traits> * tie` (`basic_ostream<_CharT, _Traits> * __tistr`)
  - `basic_ostream<_CharT, _Traits> * tie` () const
  - `void unsetf` (fmtflags \_\_mask)
  - `char_type widen` (char \_\_c) const
  - `streamsize width` () const
  - `streamsize width` (streamsize \_\_wide)
- 
- `__ostream_type` & `operator<<` (\_\_ostream\_type &(\_\_pf)(\_\_ostream\_type &))
  - `__ostream_type` & `operator<<` (\_\_ios\_type &(\_\_pf)(\_\_ios\_type &))
  - `__ostream_type` & `operator<<` (ios\_base &(\_\_pf)(ios\_base &))

### Arithmetic Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__ostream_type & operator<< (long __n)`
- `__ostream_type & operator<< (unsigned long __n)`
- `__ostream_type & operator<< (bool __n)`
- `__ostream_type & operator<< (short __n)`
- `__ostream_type & operator<< (unsigned short __n)`
- `__ostream_type & operator<< (int __n)`
- `__ostream_type & operator<< (unsigned int __n)`
- `__ostream_type & operator<< (long long __n)`
- `__ostream_type & operator<< (unsigned long long __n)`
- `__ostream_type & operator<< (double __f)`
- `__ostream_type & operator<< (float __f)`
- `__ostream_type & operator<< (long double __f)`
- `__ostream_type & operator<< (const void *__p)`
- `__ostream_type & operator<< (__streambuf_type *__sb)`

### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type & put (char_type __c)`
- `void _M_write (const char_type *__s, streamsize __n)`
- `__ostream_type & write (const char_type *__s, streamsize __n)`
- `operator void * () const`
- `bool operator! () const`



### Static Public Member Functions

- static bool `sync_with_stdio` (bool \_\_sync=true)
- static int `xalloc` () throw ()

### Static Public Attributes

- static const `fmtflags` `adjustfield`
- static const `openmode` `app`
- static const `openmode` `ate`
- static const `iostate` `badbit`
- static const `fmtflags` `basefield`
- static const `seekdir` `beg`
- static const `openmode` `binary`
- static const `fmtflags` `boolalpha`
- static const `seekdir` `cur`
- static const `fmtflags` `dec`
- static const `seekdir` `end`
- static const `iostate` `eofbit`
- static const `iostate` `failbit`
- static const `fmtflags` `fixed`
- static const `fmtflags` `floatfield`
- static const `iostate` `goodbit`
- static const `fmtflags` `hex`
- static const `openmode` `in`
- static const `fmtflags` `internal`
- static const `fmtflags` `left`
- static const `fmtflags` `oct`
- static const `openmode` `out`
- static const `fmtflags` `right`
- static const `fmtflags` `scientific`
- static const `fmtflags` `showbase`
- static const `fmtflags` `showpoint`
- static const `fmtflags` `showpos`
- static const `fmtflags` `skipws`
- static const `openmode` `trunc`
- static const `fmtflags` `unitbuf`
- static const `fmtflags` `uppercase`

### Protected Types

- enum { `_S_local_word_size` }

### Protected Member Functions

- void **\_M\_cache\_locale** (const [locale](#) &\_\_loc)
- void **\_M\_call\_callbacks** ([event](#) \_\_ev) throw ()
- void **\_M\_dispose\_callbacks** (void) throw ()
- [\\_Words](#) & **\_M\_grow\_words** (int \_\_index, bool \_\_iword)
- void **\_M\_init** () throw ()
- template<typename \_ValueT >  
    [\\_\\_ostream\\_type](#) & **\_M\_insert** (\_ValueT \_\_v)
- void **init** ([basic\\_streambuf](#)< \_CharT, \_Traits > \*\_\_sb)

### Protected Attributes

- [\\_Callback\\_list](#) \* **\_M\_callbacks**
- const [\\_\\_ctype\\_type](#) \* **\_M\_ctype**
- [iostate](#) **\_M\_exception**
- [char\\_type](#) **\_M\_fill**
- bool **\_M\_fill\_init**
- [fmtflags](#) **\_M\_flags**
- [locale](#) **\_M\_ios\_locale**
- [\\_Words](#) **\_M\_local\_word** [[\\_S\\_local\\_word\\_size](#)]
- const [\\_\\_num\\_get\\_type](#) \* **\_M\_num\_get**
- const [\\_\\_num\\_put\\_type](#) \* **\_M\_num\_put**
- [streamsize](#) **\_M\_precision**
- [basic\\_streambuf](#)< \_CharT, \_Traits > \* **\_M\_streambuf**
- [iostate](#) **\_M\_streambuf\_state**
- [basic\\_ostream](#)< \_CharT, \_Traits > \* **\_M\_tie**
- [streamsize](#) **\_M\_width**
- [\\_Words](#) \* **\_M\_word**
- int **\_M\_word\_size**
- [\\_Words](#) **\_M\_word\_zero**

### Friends

- class **sentry**

#### 5.385.1 Detailed Description

template<typename \_CharT, typename \_Traits> class std::basic\_ostream< \_CharT, \_Traits >

Controlling output.

This is the base class for all output streams. It provides text formatting of all builtin types, and communicates with any class derived from [basic\\_streambuf](#) to do the actual output.

Definition at line 57 of file ostream.

## 5.385.2 Member Typedef Documentation

**5.385.2.1** `template<typename _CharT, typename _Traits> typedef  
cctype<_CharT> std::basic_ostream< _CharT, _Traits  
>::__cctype_type`

These are non-standard types.

Reimplemented from [std::basic\\_ios< \\_CharT, \\_Traits >](#).

Definition at line 73 of file ostream.

**5.385.2.2** `template<typename _CharT, typename _Traits> typedef  
num_get<_CharT, istreambuf_iterator<_CharT, _Traits>  
> std::basic_ios< _CharT, _Traits >::__num_get_type  
[inherited]`

These are non-standard types.

Reimplemented in [std::basic\\_istream< \\_CharT, \\_Traits >](#), [std::basic\\_istream< char, \\_Traits >](#), and [std::basic\\_istream< char >](#).

Definition at line 88 of file basic\_ios.h.

**5.385.2.3** `template<typename _CharT, typename _Traits> typedef  
num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> >  
std::basic_ostream< _CharT, _Traits >::__num_put_type`

These are non-standard types.

Reimplemented from [std::basic\\_ios< \\_CharT, \\_Traits >](#).

Definition at line 72 of file ostream.

**5.385.2.4** `template<typename _CharT, typename _Traits> typedef _CharT  
std::basic_ostream< _CharT, _Traits >::__char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios< _CharT, _Traits >`.

Reimplemented in `std::basic_ofstream< _CharT, _Traits >`, `std::basic_fstream< _CharT, _Traits >`, `std::basic_iostream< _CharT, _Traits >`, `std::basic_ostringstream< _CharT, _Traits, _Alloc >`, `std::basic_stringstream< _CharT, _Traits, _Alloc >`, and `std::basic_istream< char >`.

Definition at line 61 of file `ostream`.

#### 5.385.2.5 `typedef void(* std::ios_base::event_callback)(event, ios_base &, int)` **[inherited]**

The type of an event callback function.

##### Parameters

*event* One of the members of the event enum.

*ios\_base* Reference to the `ios_base` object.

*int* The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 438 of file `ios_base.h`.

#### 5.385.2.6 `typedef _Ios_Fmtflags std::ios_base::fmtflags` **[inherited]**

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`

- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Definition at line 257 of file `ios_base.h`.

#### 5.385.2.7 `template<typename _CharT, typename _Traits> typedef _Traits::int_type std::basic_ostream<_CharT, _Traits>::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Reimplemented in `std::basic_ofstream<_CharT, _Traits>`, `std::basic_fstream<_CharT, _Traits>`, `std::basic_iostream<_CharT, _Traits>`, `std::basic_ostringstream<_CharT, _Traits, _Alloc>`, `std::basic_stringstream<_CharT, _Traits, _Alloc>`, and `std::basic_istream<char>`.

Definition at line 62 of file `ostream`.

#### 5.385.2.8 `typedef _Ios_Iostate std::ios_base::iostate [inherited]`

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 332 of file `ios_base.h`.

#### 5.385.2.9 `template<typename _CharT, typename _Traits> typedef _Traits::off_type std::basic_ostream< _CharT, _Traits >::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios< _CharT, _Traits >`.

Reimplemented in `std::basic_ofstream< _CharT, _Traits >`, `std::basic_fstream< _CharT, _Traits >`, `std::basic_iostream< _CharT, _Traits >`, `std::basic_ostringstream< _CharT, _Traits, _Alloc >`, `std::basic_stringstream< _CharT, _Traits, _Alloc >`, and `std::basic_iostream< char >`.

Definition at line 64 of file `ostream`.

#### 5.385.2.10 `typedef _Ios_Openmode std::ios_base::openmode [inherited]`

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 363 of file `ios_base.h`.

**5.385.2.11** `template<typename _CharT, typename _Traits> typedef  
_Traits::pos_type std::basic_ostream< _CharT, _Traits >::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios< _CharT, _Traits >`.

Reimplemented in `std::basic_ofstream< _CharT, _Traits >`, `std::basic_fstream< _CharT, _Traits >`, `std::basic_iostream< _CharT, _Traits >`, `std::basic_ostringstream< _CharT, _Traits, _Alloc >`, `std::basic_stringstream< _CharT, _Traits, _Alloc >`, and `std::basic_istream< char >`.

Definition at line 63 of file `ostream`.

**5.385.2.12** `typedef _Ios_Seekdir std::ios_base::seekdir [inherited]`

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 395 of file `ios_base.h`.

**5.385.2.13** `template<typename _CharT, typename _Traits> typedef _Traits  
std::basic_ostream< _CharT, _Traits >::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios< _CharT, _Traits >`.

Reimplemented in `std::basic_ofstream< _CharT, _Traits >`, `std::basic_fstream< _CharT, _Traits >`, `std::basic_iostream< _CharT, _Traits >`, `std::basic_ostringstream< _CharT, _Traits, _Alloc >`, `std::basic_stringstream< _CharT, _Traits, _Alloc >`, and `std::basic_istream< char >`.

Definition at line 65 of file `ostream`.

### 5.385.3 Member Enumeration Documentation

#### 5.385.3.1 `enum std::ios_base::event` **[inherited]**

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 421 of file `ios_base.h`.

### 5.385.4 Constructor & Destructor Documentation

#### 5.385.4.1 `template<typename _CharT, typename _Traits>` `std::basic_ostream< _CharT, _Traits >::basic_ostream (` `__streambuf_type * __sb ) [inline, explicit]`

Base constructor.

This ctor is almost never called by the user directly, rather from derived classes' initialization lists, which pass a pointer to their own stream buffer.

Definition at line 84 of file `ostream`.

#### 5.385.4.2 `template<typename _CharT, typename _Traits> virtual` `std::basic_ostream< _CharT, _Traits >::~~basic_ostream ( )` `[inline, virtual]`

Base destructor.

This does very little apart from providing a virtual base dtor.

Definition at line 93 of file `ostream`.

### 5.385.5 Member Function Documentation

#### 5.385.5.1 `const locale& std::ios_base::_M_getloc ( ) const` **[inline, inherited]**

Locale access.



**Returns**

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 708 of file ios\_base.h.

Referenced by std::money\_get< \_CharT, \_InIter >::do\_get(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_date(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::time\_put< \_CharT, \_OutIter >::do\_put(), std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::time\_put< \_CharT, \_OutIter >::put().

**5.385.5.2** `template<typename _CharT, typename _Traits> void  
std::basic_ostream< _CharT, _Traits >::_M_write ( const char_type  
* __s, streamsize __n ) [inline]`

Simple insertion.

**Parameters**

*c* The character to insert.

**Returns**

\*this

Tries to insert *c*.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 289 of file ostream.

Referenced by std::basic\_ostream< \_CharT, \_Traits >::write().

**5.385.5.3** `template<typename _CharT, typename _Traits> bool std::basic_ios<  
_CharT, _Traits >::bad ( ) const [inline, inherited]`

Fast error checking.

**Returns**

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 203 of file `basic_ios.h`.

**5.385.5.4** `template<typename _CharT, typename _Traits> void  
std::basic_ios< _CharT, _Traits >::clear ( iostate __state = goodbit )  
[inherited]`

[Re]sets the error state.

**Parameters**

*state* The new state flag(s) to set.

See [std::ios\\_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

Definition at line 42 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::exceptions()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::rdstate()`.

Referenced by `std::basic_ios< char, _Traits >::exceptions()`, `std::basic_fstream< _CharT, _Traits >::open()`, `std::basic_ofstream< _CharT, _Traits >::open()`, `std::basic_ifstream< _CharT, _Traits >::open()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ios< char, _Traits >::setstate()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

**5.385.5.5** `template<typename _CharT, typename _Traits> basic_ios<  
_CharT, _Traits> & std::basic_ios< _CharT, _Traits >::copyfmt (  
const basic_ios< _CharT, _Traits> & __rhs ) [inherited]`

Copies fields of `__rhs` into this.

**Parameters**

*\_\_rhs* The source values for the copies.

**Returns**

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy `exceptions()`.

Definition at line 64 of file `basic_ios.tcc`.

References `std::basic_ios< _CharT, _Traits >::exceptions()`, `std::basic_ios< _CharT, _Traits >::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios< _CharT, _Traits >::tie()`, and `std::ios_base::width()`.

#### 5.385.5.6 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::eof( ) const [inline, inherited]`

Fast error checking.

**Returns**

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 182 of file `basic_ios.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

#### 5.385.5.7 `template<typename _CharT, typename _Traits> iostate std::basic_ios< _CharT, _Traits >::exceptions( ) const [inline, inherited]`

Throwing exceptions on errors.

**Returns**

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of [exceptions\(iostate\)](#) for the meaning of the return value.

Definition at line 214 of file basic\_ios.h.

Referenced by std::basic\_ios< \_CharT, \_Traits >::clear(), and std::basic\_ios< \_CharT, \_Traits >::copyfmt().

**5.385.5.8** `template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::exceptions ( iostate __except ) [inline, inherited]`

Throwing exceptions on errors.

#### Parameters

*except* The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type [std::ios\\_base::failure](#) is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
 std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

 std::ifstream f ("/etc/motd");

 std::cerr << "Setting badbit\n";
 f.setstate (std::ios_base::badbit);

 std::cerr << "Setting exception mask\n";
 f.exceptions (std::ios_base::badbit);
}
```

Definition at line 249 of file basic\_ios.h.

**5.385.5.9** `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::fail ( ) const [inline, inherited]`

Fast error checking.

**Returns**

True if either the badbit or the failbit is set.

Checking the badbit in [fail\(\)](#) is historical practice. Note that other iostate flags may also be set.

Definition at line 193 of file basic\_ios.h.

Referenced by std::basic\_ios< char, \_Traits >::operator void \*(), std::basic\_ios< char, \_Traits >::operator!(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_ostream< \_CharT, \_Traits >::tellp(), and std::regex\_traits< \_Ch\_type >::value().

**5.385.5.10** `template<typename _CharT, typename _Traits> char_type  
std::basic_ios< _CharT, _Traits >::fill ( ) const [inline,  
inherited]`

Retrieves the *empty* character.

**Returns**

The current fill character.

It defaults to a space ( ' ' ) in the current locale.

Definition at line 362 of file basic\_ios.h.

Referenced by std::basic\_ios< \_CharT, \_Traits >::copyfmt(), and std::basic\_ios< char, \_Traits >::fill().

**5.385.5.11** `template<typename _CharT, typename _Traits> char_type  
std::basic_ios< _CharT, _Traits >::fill ( char_type __ch )  
[inline, inherited]`

Sets a new *empty* character.

**Parameters**

*ch* The new character.

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and  $Q < P$ . It defaults to a space ( ' ') in the current locale.

Definition at line 382 of file `basic_ios.h`.

#### 5.385.5.12 `fmtflags std::ios_base::flags ( ) const` [inline, inherited]

Access to format flags.

##### Returns

The format control flags for both input and output.

Definition at line 553 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, `std::operator>>()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

#### 5.385.5.13 `fmtflags std::ios_base::flags ( fmtflags __fmtfl )` [inline, inherited]

Setting new format flags all at once.

##### Parameters

*fmtfl* The new flags to set.

##### Returns

The previous format control flags.

This function overwrites all the format flags with *fmtfl*.

Definition at line 564 of file `ios_base.h`.

#### 5.385.5.14 `template<typename _CharT, typename _Traits> basic_ostream< _CharT, _Traits> & std::basic_ostream< _CharT, _Traits>::flush ( )`

Synchronizing the stream buffer.

**Returns**

`*this`

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf() ->pubsync()`, and if that returns -1, sets badbit.

Definition at line 213 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

Referenced by `std::flush()`.

**5.385.5.15 `std::ios_base::getloc()` const [inline, inherited]**

Locale access.

**Returns**

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 697 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::money_put< _CharT, _OutIter >::do_put()`, `std::basic_ios< _CharT, _Traits >::imbue()`, `std::operator>>()`, and `std::ws()`.

**5.385.5.16 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::good()` const [inline, inherited]**

Fast error checking.

**Returns**

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 172 of file `basic_ios.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

**5.385.5.17** `template<typename _CharT, typename _Traits > locale  
std::basic_ios< _CharT, _Traits >::imbue ( const locale & __loc )  
[inherited]`

Moves to a new locale.

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional 110n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Reimplemented from `std::ios_base`.

Definition at line 115 of file `basic_ios.tcc`.

References `std::ios_base::getloc()`, and `std::basic_ios< _CharT, _Traits >::rdbuf()`.

Referenced by `std::operator<<()`.

**5.385.5.18** `template<typename _CharT, typename _Traits> void  
std::basic_ios< _CharT, _Traits >::init ( basic_streambuf<  
_CharT, _Traits > * __sb ) [protected, inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 127 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

Referenced by `std::basic_fstream< _CharT, _Traits >::basic_fstream()`, `std::basic_ifstream< _CharT, _Traits >::basic_ifstream()`, `std::basic_ios< char, _Traits >::basic_ios()`, `std::basic_istream< char >::basic_istream()`, `std::basic_istreamstream< _CharT, _Traits, _Alloc >::basic_istreamstream()`, `std::basic_ofstream< _CharT, _Traits >::basic_ofstream()`, `std::basic_ostream< char >::basic_ostream()`, `std::basic_ostringstream< _CharT, _Traits, _Alloc >::basic_ostringstream()`, and `std::basic_stringstream< _CharT, _Traits, _Alloc >::basic_stringstream()`.



**5.385.5.19 long& std::ios\_base::iword ( int \_\_ix ) [inline, inherited]**

Access to integer array.

**Parameters**

*\_\_ix* Index into the array.

**Returns**

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 743 of file ios\_base.h.

**5.385.5.20 template<typename \_CharT, typename \_Traits> char  
std::basic\_ios< \_CharT, \_Traits >::narrow ( char\_type \_\_c, char  
\_\_dfault ) const [inline, inherited]**

Squeezes characters.

**Parameters**

*c* The character to narrow.

*dfault* The character to narrow.

**Returns**

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c,dfault)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.ht>

Definition at line 422 of file basic\_ios.h.

**5.385.5.21** `template<typename _CharT, typename _Traits> std::basic_ios<_CharT, _Traits>::operator void * ( ) const [inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 113 of file `basic_ios.h`.

**5.385.5.22** `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::operator! ( ) const [inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 117 of file `basic_ios.h`.

**5.385.5.23** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< ( __ios_type &(*)(__ios_type &) __pf ) [inline]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

Definition at line 119 of file `ostream`.

**5.385.5.24** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< ( unsigned short __n ) [inline]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 182 of file ostream.

**5.385.5.25** `template<typename _CharT, typename _Traits > basic_ostream<  
_CharT, _Traits > & std::basic_ostream< _CharT, _Traits  
>::operator<< ( __streambuf_type * __sb )`

Extracting from another streambuf.

**Parameters**

*sb* A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from *sb* and inserted into \*this until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from *sb*, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 122 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.385.5.26** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream< _CharT, _Traits >::operator<< ( long __n )  
[inline]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 167 of file `ostream`.

```
5.385.5.27 template<typename _CharT, typename _Traits > basic_ostream<
 _CharT, _Traits > & std::basic_ostream< _CharT, _Traits
 >::operator<< (int __n)
```

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 108 of file `ostream.tcc`.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

```
5.385.5.28 template<typename _CharT, typename _Traits> __ostream_type&
 std::basic_ostream< _CharT, _Traits >::operator<< (
 __ostream_type &(*)(__ostream_type &) __pf) [inline]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

Definition at line 110 of file `ostream`.

**5.385.5.29** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream< _CharT, _Traits >::operator<< ( unsigned int  
__n ) [inline]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 193 of file ostream.

**5.385.5.30** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream< _CharT, _Traits >::operator<< ( unsigned  
long __n ) [inline]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 171 of file ostream.

**5.385.5.31** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream< _CharT, _Traits >::operator<< ( long long  
__n ) [inline]`

Basic arithmetic inserters.

### Parameters

*A* variable of builtin type.

### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 202 of file `ostream`.

**5.385.5.32** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< ( unsigned long long __n ) [inline]`

Basic arithmetic inserters.

### Parameters

*A* variable of builtin type.

### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 206 of file `ostream`.

**5.385.5.33** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< ( bool __n ) [inline]`

Basic arithmetic inserters.

### Parameters

*A* variable of builtin type.

### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 175 of file `ostream`.

**5.385.5.34** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream<_CharT, _Traits>::operator<< ( const void *  
__p ) [inline]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 227 of file `ostream`.

**5.385.5.35** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream<_CharT, _Traits>::operator<< ( double __f  
) [inline]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 211 of file `ostream`.

**5.385.5.36** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream< _CharT, _Traits >::operator<< ( ios_base  
&(*)(ios_base &) __pf ) [inline]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the `iomanip` header.

Definition at line 129 of file `ostream`.

**5.385.5.37** `template<typename _CharT, typename _Traits> basic_ostream<  
_CharT, _Traits > & std::basic_ostream< _CharT, _Traits  
>::operator<< ( short __n )`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

\**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 94 of file `ostream.tcc`.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

**5.385.5.38** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream< _CharT, _Traits >::operator<< ( float __f )  
[inline]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.



### Returns

\*this if successful

These functions use the stream's current locale (specifically, the [num\\_get](#) facet) to perform numeric formatting.

Definition at line 215 of file ostream.

**5.385.5.39** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream< _CharT, _Traits >::operator<< ( long double  
__f ) [inline]`

Basic arithmetic inserters.

### Parameters

*A* variable of builtin type.

### Returns

\*this if successful

These functions use the stream's current locale (specifically, the [num\\_get](#) facet) to perform numeric formatting.

Definition at line 223 of file ostream.

**5.385.5.40** `streamsize std::ios_base::precision ( streamsize __prec )  
[inline, inherited]`

Changing flags.

### Parameters

*prec* The new precision value.

### Returns

The previous value of [precision\(\)](#).

Definition at line 632 of file ios\_base.h.

**5.385.5.41** streamsize std::ios\_base::precision ( ) const [inline, inherited]

Flags access.

#### Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 623 of file ios\_base.h.

Referenced by std::basic\_ios< \_CharT, \_Traits >::copyfmt(), and std::operator<<().

**5.385.5.42** template<typename \_CharT, typename \_Traits > basic\_ostream< \_CharT, \_Traits > & std::basic\_ostream< \_CharT, \_Traits >::put ( char\_type \_\_c )

Simple insertion.

#### Parameters

*c* The character to insert.

#### Returns

\*this

Tries to insert *c*.

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 151 of file ostream.tcc.

References std::ios\_base::badbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), and std::basic\_ios< \_CharT, \_Traits >::setstate().

Referenced by std::endl(), and std::ends().

**5.385.5.43** void\*& std::ios\_base::pword ( int \_\_ix ) [inline, inherited]

Access to void pointer array.

#### Parameters

`__ix` Index into the array.

#### Returns

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 764 of file `ios_base.h`.

**5.385.5.44** `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rdbuf ( basic_streambuf<_CharT, _Traits> * __sb ) [inherited]`

Changing the underlying buffer.

#### Parameters

`sb` The new stream buffer.

#### Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream foo; // or some other derived type
std::streambuf* p =;

foo.ios::rdbuf(p); // ios == basic_ios<char>
```

Definition at line 54 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::clear()`.

**5.385.5.45** `template<typename _CharT, typename _Traits>  
 basic_streambuf<_CharT, _Traits>* std::basic_ios< _CharT,  
 _Traits >::rdbuf ( ) const [inline, inherited]`

Accessing the underlying buffer.

#### Returns

The current stream buffer.

This does not change the state of the stream.

Reimplemented in [std::basic\\_ifstream< \\_CharT, \\_Traits >](#), [std::basic\\_ofstream< \\_CharT, \\_Traits >](#), [std::basic\\_fstream< \\_CharT, \\_Traits >](#), [std::basic\\_istream< \\_CharT, \\_Traits, \\_Alloc >](#), [std::basic\\_ostringstream< \\_CharT, \\_Traits, \\_Alloc >](#), and [std::basic\\_stringstream< \\_CharT, \\_Traits, \\_Alloc >](#).

Definition at line 313 of file `basic_ios.h`.

Referenced by `std::basic_ostream< char >::_M_write()`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ios< _CharT, _Traits >::imbue()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

**5.385.5.46** `template<typename _CharT, typename _Traits> iostate  
 std::basic_ios< _CharT, _Traits >::rdstate ( ) const [inline,  
 inherited]`

Returns the error state of the stream buffer.

#### Returns

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 129 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::basic_ios< char, _Traits >::eof()`, `std::basic_ios< char, _Traits >::fail()`, `std::basic_ios< char, _Traits >::good()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ios< char, _Traits >::setstate()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

#### 5.385.5.47 `void std::ios_base::register_callback ( event_callback __fn, int __index ) [inherited]`

Add the callback `__fn` with parameter `__index`.

##### Parameters

`__fn` The function to add.

`__index` The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

#### 5.385.5.48 `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp ( pos_type __pos )`

Changing the current write position.

##### Parameters

`pos` A file position object.

##### Returns

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.

Definition at line 260 of file `ostream.tcc`.

References std::ios\_base::badbit, std::basic\_ios< \_CharT, \_Traits >::fail(), std::ios\_base::failbit, std::ios\_base::goodbit, std::ios\_base::out, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), and std::basic\_ios< \_CharT, \_Traits >::setstate().

**5.385.5.49** `template<typename _CharT, typename _Traits > basic_ostream<  
_CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp  
( off_type __off, ios_base::seekdir __dir )`

Changing the current write position.

#### Parameters

*off* A file offset object.

*dir* The direction in which to seek.

#### Returns

\*this

If `fail()` is not true, calls `rdbuf() -> pubseekoff(off, dir)`. If that function fails, sets failbit.

Definition at line 292 of file ostream.tcc.

References std::ios\_base::badbit, std::basic\_ios< \_CharT, \_Traits >::fail(), std::ios\_base::failbit, std::ios\_base::goodbit, std::ios\_base::out, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), and std::basic\_ios< \_CharT, \_Traits >::setstate().

**5.385.5.50** `fmtflags std::ios_base::setf( fmtflags __fmtfl, fmtflags __mask )  
[inline, inherited]`

Setting new format flags.

#### Parameters

*fmtfl* Additional flags to set.

*mask* The flags mask for *fmtfl*.

#### Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 597 of file ios\_base.h.

**5.385.5.51** `fmtflags std::ios_base::setf ( fmtflags __fmtfl ) [inline, inherited]`

Setting new format flags.

**Parameters**

*fmtfl* Additional flags to set.

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 580 of file ios\_base.h.

Referenced by `std::dec()`, `std::fixed()`, `std::hex()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

**5.385.5.52** `template<typename _CharT, typename _Traits> void  
std::basic_ios< _CharT, _Traits >::setstate ( iostate __state )  
[inline, inherited]`

Sets additional flags in the error state.

**Parameters**

*state* The additional state flag(s) to set.

See [std::ios\\_base::iostate](#) for the possible bit values.

Definition at line 149 of file basic\_ios.h.

Referenced by `std::basic_ostream< char >::_M_write()`, `std::basic_fstream< _CharT, _Traits >::close()`, `std::basic_ofstream< _CharT, _Traits >::close()`, `std::basic_ifstream< _CharT, _Traits >::close()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_fstream< _CharT, _Traits >::open()`, `std::basic_ofstream< _CharT, _Traits >::open()`, `std::basic_ifstream< _CharT, _Traits >::open()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`,

std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::sync(), std::basic\_istream< \_CharT, \_Traits >::unget(), and std::ws().

**5.385.5.53** static bool std::ios\_base::sync\_with\_stdio ( bool \_\_sync = true )  
[static, inherited]

Interaction with the standard C I/O objects.

#### Parameters

*sync* Whether to synchronize or not.

#### Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

**5.385.5.54** template<typename \_CharT, typename \_Traits > basic\_ostream< \_CharT, \_Traits >::pos\_type std::basic\_ostream< \_CharT, \_Traits >::tellp ( )

Getting the current write position.

#### Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rddbuf()->pubseekoff(0, cur, out)`.

Definition at line 239 of file ostream.tcc.

References std::ios\_base::badbit, std::ios\_base::cur, std::basic\_ios< \_CharT, \_Traits >::fail(), std::ios\_base::out, and std::basic\_ios< \_CharT, \_Traits >::rddbuf().



**5.385.5.55** `template<typename _CharT, typename _Traits>  
 basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits  
 >::tie( ) const [inline, inherited]`

Fetches the current *tied* stream.

#### Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 287 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

**5.385.5.56** `template<typename _CharT, typename _Traits>  
 basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits  
 >::tie( basic_ostream<_CharT, _Traits > * __tiestr ) [inline,  
 inherited]`

Ties this stream to an output stream.

#### Parameters

*tiestr* The output stream.

#### Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 299 of file `basic_ios.h`.

**5.385.5.57** `void std::ios_base::unsetf( fmtflags __mask ) [inline,  
 inherited]`

Clearing format flags.

**Parameters**

*mask* The flags to unset.

This function clears *mask* in the format flags.

Definition at line 612 of file ios\_base.h.

Referenced by std::noboolalpha(), std::noshowbase(), std::noshowpoint(), std::noshowpos(), std::noskipws(), std::nounitbuf(), and std::nouppercase().

**5.385.5.58** `template<typename _CharT, typename _Traits> char_type  
std::basic_ios< _CharT, _Traits >::widen ( char __c ) const  
[inline, inherited]`

Widens characters.

**Parameters**

*c* The character to widen.

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 441 of file basic\_ios.h.

Referenced by std::endl(), std::basic\_ios< char, \_Traits >::fill(), std::basic\_istream< char >::get(), std::basic\_istream< char >::getline(), std::getline(), and std::operator>>().

**5.385.5.59** `streamsize std::ios_base::width ( ) const [inline,  
inherited]`

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 646 of file ios\_base.h.

Referenced by std::basic\_ios< \_CharT, \_Traits >::copyfmt(), std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::operator<>().

#### 5.385.5.60 streamsize std::ios\_base::width ( streamsize \_\_wide ) [inline, inherited]

Changing flags.

##### Parameters

*wide* The new width value.

##### Returns

The previous value of [width\(\)](#).

Definition at line 655 of file ios\_base.h.

#### 5.385.5.61 template<typename \_CharT, typename \_Traits > basic\_ostream< \_CharT, \_Traits > & std::basic\_ostream< \_CharT, \_Traits >::write ( const char\_type \* \_\_s, streamsize \_\_n )

Character string insertion.

##### Parameters

*s* The array to insert.

*n* Maximum number of characters to insert.

##### Returns

\*this

Characters are copied from *s* and inserted into the stream until one of the following happens:

- *n* characters are inserted
- inserting into the output sequence fails (in this case, badbit will be set in the stream's error state)

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 185 of file ostream.tcc.

References std::basic\_ostream< \_CharT, \_Traits >::\_M\_write(), and std::ios\_base::badbit.

**5.385.5.62 static int std::ios\_base::xalloc ( ) throw () [static, inherited]**

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the iword and pword functions. The expectation is that an application calls xalloc in order to obtain an index in the iword and pword arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. xalloc is guaranteed to return an index that is safe to use in the iword and pword arrays.

**5.385.6 Member Data Documentation****5.385.6.1 const fmtflags std::ios\_base::adjustfield [static, inherited]**

A mask of left|right|internal. Useful for the 2-arg form of setf.

Definition at line 312 of file ios\_base.h.

Referenced by std::num\_put< \_CharT, \_OutIter >::do\_put(), std::internal(), std::left(), and std::right().

**5.385.6.2 const openmode std::ios\_base::app [static, inherited]**

Seek to end before each write.

Definition at line 366 of file ios\_base.h.

**5.385.6.3 `const openmode std::ios_base::ate` [`static`, `inherited`]**

Open and seek to end immediately after opening.

Definition at line 369 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

**5.385.6.4 `const iostate std::ios_base::badbit` [`static`, `inherited`]**

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 336 of file `ios_base.h`.

Referenced by `std::basic_ostream< char >::_M_write()`, `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::basic_ios< char, _Traits >::fail()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ios< _CharT, _Traits >::init()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, `std::basic_istream< _CharT, _Traits >::unget()`, `std::basic_ostream< _CharT, _Traits >::write()`, and `std::basic_ostream< _CharT, _Traits >::sentry::~sentry()`.

**5.385.6.5 `const fmtflags std::ios_base::basefield` [`static`, `inherited`]**

A mask of `dec|oct|hex`. Useful for the 2-arg form of `setf`.

Definition at line 315 of file `ios_base.h`.

Referenced by `std::dec()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::hex()`, `std::oct()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

**5.385.6.6 const seekdir std::ios\_base::beg [static, inherited]**

Request a seek relative to the beginning of the stream.

Definition at line 398 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::seekpos().

**5.385.6.7 const openmode std::ios\_base::binary [static, inherited]**

Perform input and output in binary mode (as opposed to text mode). /// This is probably not what you think it is; see /// <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 374 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::showmanyc().

**5.385.6.8 const fmtflags std::ios\_base::boolalpha [static, inherited]**

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 260 of file ios\_base.h.

Referenced by std::boolalpha(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::noboolalpha().

**5.385.6.9 const seekdir std::ios\_base::cur [static, inherited]**

Request a seek relative to the current position within the sequence.

Definition at line 401 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::imbue(), std::basic\_filebuf< \_CharT, \_Traits >::overflow(), std::basic\_filebuf< \_CharT, \_Traits >::pbackfail(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_filebuf< \_CharT, \_Traits >::seekoff(), std::basic\_istream< \_CharT, \_Traits >::tellg(), and std::basic\_ostream< \_CharT, \_Traits >::tellp().

**5.385.6.10 const fmtflags std::ios\_base::dec [static, inherited]**

Converts integer input or generates integer output in decimal base.

Definition at line 263 of file ios\_base.h.

Referenced by std::dec().

**5.385.6.11 const seekdir std::ios\_base::end [static, inherited]**

Request a seek relative to the current end of the sequence.

Definition at line 404 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::open(), and std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff().

**5.385.6.12 const iostate std::ios\_base::eofbit [static, inherited]**

Indicates that an input operation reached the end of an input sequence.

Definition at line 339 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_date(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::eof(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::unget(), and std::ws().

**5.385.6.13 const iostate std::ios\_base::failbit [static, inherited]**

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 344 of file ios\_base.h.

Referenced by std::basic\_fstream< \_CharT, \_Traits >::close(), std::basic\_ofstream< \_CharT, \_Traits >::close(), std::basic\_ifstream< \_CharT, \_Traits >::close(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::fail(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_fstream< \_CharT, \_Traits >::open(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry(), and std::basic\_istream< \_CharT, \_Traits >::sentry::sentry().

#### 5.385.6.14 const fmtflags std::ios\_base::fixed [static, inherited]

Generate floating-point output in fixed-point notation.

Definition at line 266 of file ios\_base.h.

Referenced by std::fixed().

#### 5.385.6.15 const fmtflags std::ios\_base::floatfield [static, inherited]

A mask of scientific|fixed. Useful for the 2-arg form of setf.

Definition at line 318 of file ios\_base.h.

Referenced by std::fixed(), and std::scientific().

#### 5.385.6.16 const iostate std::ios\_base::goodbit [static, inherited]

Indicates all is well.

Definition at line 347 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(),



`std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ios< _CharT, _Traits >::init()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::basic_istream< _CharT, _Traits >::sync()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

#### 5.385.6.17 `const fmtflags std::ios_base::hex` [static, inherited]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 269 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::hex()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

#### 5.385.6.18 `const openmode std::ios_base::in` [static, inherited]

Open for input. Default for `ifstream` and `fstream`.

Definition at line 377 of file `ios_base.h`.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_set_buffer()`, `std::basic_ifstream< _CharT, _Traits >::open()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

#### 5.385.6.19 `const fmtflags std::ios_base::internal` [static, inherited]

Adds fill characters at a designated internal point in certain `///` generated output, or identical to `right` if no such point is `///` designated.

Definition at line 274 of file `ios_base.h`.

Referenced by `std::internal()`.

**5.385.6.20** `const fmtflags std::ios_base::left` [`static`, `inherited`]

Adds fill characters on the right (final positions) of certain /// generated output. (I.e., the thing you print is flush left.).

Definition at line 278 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, and `std::left()`.

**5.385.6.21** `const fmtflags std::ios_base::oct` [`static`, `inherited`]

Converts integer input or generates integer output in octal base.

Definition at line 281 of file `ios_base.h`.

Referenced by `std::oct()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

**5.385.6.22** `const openmode std::ios_base::out` [`static`, `inherited`]

Open for output. Default for `ofstream` and `fstream`.

Definition at line 380 of file `ios_base.h`.

Referenced by `std::basic_filebuf< char_type, traits_type >::M_set_buffer()`, `std::basic_ofstream< _CharT, _Traits >::open()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

**5.385.6.23** `const fmtflags std::ios_base::right` [`static`, `inherited`]

Adds fill characters on the left (initial positions) of certain /// generated output. (I.e., the thing you print is flush right.).

Definition at line 285 of file `ios_base.h`.

Referenced by std::right().

**5.385.6.24 const fmtflags std::ios\_base::scientific [static, inherited]**

Generates floating-point output in scientific notation.

Definition at line 288 of file ios\_base.h.

Referenced by std::scientific().

**5.385.6.25 const fmtflags std::ios\_base::showbase [static, inherited]**

Generates a prefix indicating the numeric base of generated integer /// output.

Definition at line 292 of file ios\_base.h.

Referenced by std::noshowbase(), and std::showbase().

**5.385.6.26 const fmtflags std::ios\_base::showpoint [static, inherited]**

Generates a decimal-point character unconditionally in generated /// floating-point output.

Definition at line 296 of file ios\_base.h.

Referenced by std::noshowpoint(), and std::showpoint().

**5.385.6.27 const fmtflags std::ios\_base::showpos [static, inherited]**

Generates a + sign in non-negative generated numeric output.

Definition at line 299 of file ios\_base.h.

Referenced by std::noshowpos(), and std::showpos().

**5.385.6.28 const fmtflags std::ios\_base::skipws [static, inherited]**

Skips leading white space before certain input operations.

Definition at line 302 of file `ios_base.h`.

Referenced by `std::noskipws()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, and `std::skipws()`.

#### 5.385.6.29 `const openmode std::ios_base::trunc` [static, inherited]

Open for input. Default for `ofstream`.

Definition at line 383 of file `ios_base.h`.

#### 5.385.6.30 `const fmtflags std::ios_base::unitbuf` [static, inherited]

Flushes output after each output operation.

Definition at line 305 of file `ios_base.h`.

Referenced by `std::nunitbuf()`, `std::unitbuf()`, and `std::basic_ostream< _CharT, _Traits >::sentry::~sentry()`.

#### 5.385.6.31 `const fmtflags std::ios_base::uppercase` [static, inherited]

Replaces certain lowercase letters with their uppercase equivalents /// in generated output.

Definition at line 309 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, `std::nouppercase()`, and `std::uppercase()`.

The documentation for this class was generated from the following files:

- [ostream](#)
- [ostream.tcc](#)

### 5.386 `std::basic_ostream< _CharT, _Traits >::sentry` Class Reference

Performs setup work for output streams.

## Public Member Functions

- [sentry](#) ([basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os)
- [~sentry](#) ()
- [operator bool](#) () const

### 5.386.1 Detailed Description

**template<typename \_CharT, typename \_Traits> class std::basic\_ostream< \_CharT, \_Traits >::sentry**

Performs setup work for output streams. Objects of this class are created before all of the standard inserters are run. It is responsible for *exception-safe prefix and suffix operations*.

Definition at line 379 of file ostream.

### 5.386.2 Constructor & Destructor Documentation

**5.386.2.1** **template<typename \_CharT, typename \_Traits>**  
**std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry (**  
**basic\_ostream< \_CharT, \_Traits > & \_\_os ) [explicit]**

The constructor performs preparatory work.

#### Parameters

*os* The output stream to guard.

If the stream state is good (*os.good()* is true), then if the stream is tied to another output stream, *is.tie()* -> [flush\(\)](#) is called to synchronize the output sequences.

If the stream state is still good, then the sentry state becomes true (*okay*).

Definition at line 49 of file ostream.tcc.

References [std::ios\\_base::failbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::good\(\)](#), [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#), and [std::basic\\_ios< \\_CharT, \\_Traits >::tie\(\)](#).

**5.386.2.2** **template<typename \_CharT, typename \_Traits>**  
**std::basic\_ostream< \_CharT, \_Traits >::sentry::~sentry ( )**  
**[inline]**

## 5.387 `std::basic_ostringstream<_CharT, _Traits, _Alloc>` Class Template Reference 2107

---

Possibly flushes the stream.

If `ios_base::unitbuf` is set in `os.flags()`, and `std::uncaught_exception()` is true, the sentry destructor calls `flush()` on the output stream.

Definition at line 407 of file `ostream`.

References `std::ios_base::badbit`, `std::uncaught_exception()`, and `std::ios_base::unitbuf`.

### 5.386.3 Member Function Documentation

**5.386.3.1** `template<typename _CharT, typename _Traits>`  
`std::basic_ostream<_CharT, _Traits>::sentry::operator bool ( )`  
`const [inline, explicit]`

Quick status checking.

#### Returns

The sentry state.

For ease of use, sentries may be converted to booleans. The return value is that of the sentry state (`true == okay`).

Definition at line 428 of file `ostream`.

The documentation for this class was generated from the following files:

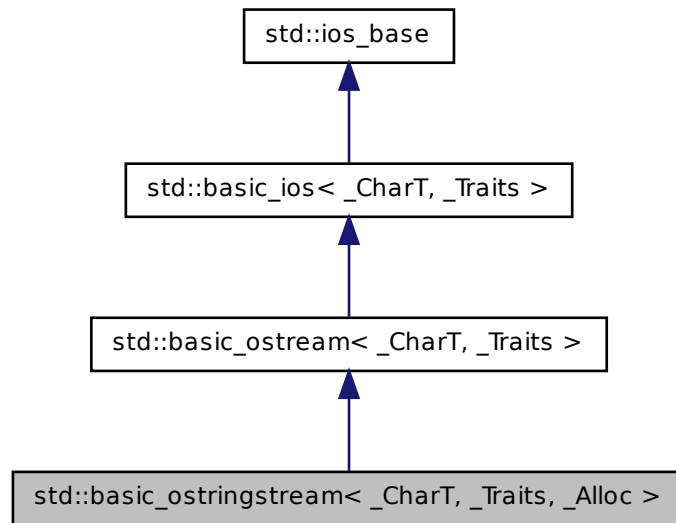
- `ostream`
- `ostream.tcc`

## 5.387 `std::basic_ostringstream<_CharT, _Traits, _Alloc>` Class Template Reference

Controlling output for `std::string`.

This class supports writing to objects of type `std::basic_string`, using the inherited functions from `std::basic_ostream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

Inheritance diagram for `std::basic_ostringstream<_CharT, _Traits, _Alloc>`:



### Public Types

- typedef `ctype<_CharT>` `__ctype_type`
- typedef `basic_ios<_CharT, _Traits>` `__ios_type`
- typedef `num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>>` `__num_put_type`
- typedef `basic_ostream<char_type, traits_type>` `__ostream_type`
- typedef `basic_streambuf<_CharT, _Traits>` `__streambuf_type`
- typedef `basic_string<_CharT, _Traits, _Alloc>` `__string_type`
- typedef `basic_stringbuf<_CharT, _Traits, _Alloc>` `__stringbuf_type`
- typedef `_Alloc` `allocator_type`
- typedef `_CharT` `char_type`
- enum `event` { `erase_event`, `imbue_event`, `copyfmt_event` }
- typedef `void(* event_callback)(event, ios_base &, int)`
- typedef `_Ios_Fmtflags` `fmtflags`
- typedef `traits_type::int_type` `int_type`
- typedef `int` `io_state`

- typedef `_Ios_Iostate` [iostate](#)
  - typedef `traits_type::off_type` [off\\_type](#)
  - typedef `int` [open\\_mode](#)
  - typedef `_Ios_Openmode` [openmode](#)
  - typedef `traits_type::pos_type` [pos\\_type](#)
  - typedef `int` [seek\\_dir](#)
  - typedef `_Ios_Seekdir` [seekdir](#)
  - typedef `std::streamoff` [streamoff](#)
  - typedef `std::streampos` [streampos](#)
  - typedef `_Traits` [traits\\_type](#)
- 
- typedef `num_get<_CharT, istreambuf_iterator<_CharT, _Traits>>` [\\_\\_num\\_get\\_type](#)

#### Public Member Functions

- [basic\\_ostringstream](#) ([ios\\_base::openmode](#) \_\_mode=[ios\\_base::out](#))
- [basic\\_ostringstream](#) (const [\\_\\_string\\_type](#) &\_\_str, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::out](#))
- [~basic\\_ostringstream](#) ()
- const [locale](#) & [\\_M\\_getloc](#) () const
- void [\\_M\\_setstate](#) ([iostate](#) \_\_state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) \_\_state=[goodbit](#))
- [basic\\_ios](#) & [copyfmt](#) (const [basic\\_ios](#) &\_\_rhs)
- bool [eof](#) () const
- [iostate](#) [exceptions](#) () const
- void [exceptions](#) ([iostate](#) \_\_except)
- bool [fail](#) () const
- [char\\_type](#) [fill](#) () const
- [char\\_type](#) [fill](#) ([char\\_type](#) \_\_ch)
- [fmtflags](#) [flags](#) ([fmtflags](#) \_\_fmtfl)
- [fmtflags](#) [flags](#) () const
- [\\_\\_ostream\\_type](#) & [flush](#) ()
- [locale](#) [getloc](#) () const
- bool [good](#) () const
- [locale](#) [imbue](#) (const [locale](#) &\_\_loc)
- long & [iword](#) (int \_\_ix)
- char [narrow](#) ([char\\_type](#) \_\_c, char \_\_dfault) const
- [streamsize](#) [precision](#) () const
- [streamsize](#) [precision](#) ([streamsize](#) \_\_prec)
- void \*& [pword](#) (int \_\_ix)



- `__stringbuf_type * rdbuf () const`
  - `basic_streambuf<_CharT, _Traits> * rdbuf (basic_streambuf<_CharT, _Traits> * __sb)`
  - `iostate rdstate () const`
  - `void register_callback (event_callback __fn, int __index)`
  - `__ostream_type & seekp (pos_type)`
  - `__ostream_type & seekp (off_type, ios_base::seekdir)`
  - `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
  - `fmtflags setf (fmtflags __fmtfl)`
  - `void setstate (iostate __state)`
  - `void str (const __string_type & __s)`
  - `__string_type str () const`
  - `pos_type tellp ()`
  - `basic_ostream<_CharT, _Traits> * tie (basic_ostream<_CharT, _Traits> * __tistr)`
  - `basic_ostream<_CharT, _Traits> * tie () const`
  - `void unsetf (fmtflags __mask)`
  - `char_type widen (char __c) const`
  - `streamsize width () const`
  - `streamsize width (streamsize __wide)`
- 
- `__ostream_type & operator<< (__ostream_type &(__pf)(__ostream_type &))`
  - `__ostream_type & operator<< (__ios_type &(__pf)(__ios_type &))`
  - `__ostream_type & operator<< (ios_base &(__pf)(ios_base &))`

### Arithmetic Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__ostream_type & operator<< (long __n)`
- `__ostream_type & operator<< (unsigned long __n)`
- `__ostream_type & operator<< (bool __n)`
- `__ostream_type & operator<< (short __n)`
- `__ostream_type & operator<< (unsigned short __n)`
- `__ostream_type & operator<< (int __n)`
- `__ostream_type & operator<< (unsigned int __n)`

- `__ostream_type & operator<< (long long __n)`
- `__ostream_type & operator<< (unsigned long long __n)`
- `__ostream_type & operator<< (double __f)`
- `__ostream_type & operator<< (float __f)`
- `__ostream_type & operator<< (long double __f)`
- `__ostream_type & operator<< (const void *__p)`
- `__ostream_type & operator<< (__streambuf_type *__sb)`

### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type & put (char_type __c)`
- `void _M_write (const char_type *__s, streamsize __n)`
- `__ostream_type & write (const char_type *__s, streamsize __n)`
- `operator void * () const`
- `bool operator! () const`

### Static Public Member Functions

- static bool `sync_with_stdio` (bool `__sync=true`)
- static int `xalloc` () throw ()

### Static Public Attributes

- static const `fmtflags adjustfield`
- static const `openmode app`
- static const `openmode ate`
- static const `iostate badbit`
- static const `fmtflags basefield`
- static const `seekdir beg`
- static const `openmode binary`
- static const `fmtflags boolalpha`
- static const `seekdir cur`
- static const `fmtflags dec`

- static const [seekdir](#) end
- static const [iostate](#) eofbit
- static const [iostate](#) failbit
- static const [fmtflags](#) fixed
- static const [fmtflags](#) floatfield
- static const [iostate](#) goodbit
- static const [fmtflags](#) hex
- static const [openmode](#) in
- static const [fmtflags](#) internal
- static const [fmtflags](#) left
- static const [fmtflags](#) oct
- static const [openmode](#) out
- static const [fmtflags](#) right
- static const [fmtflags](#) scientific
- static const [fmtflags](#) showbase
- static const [fmtflags](#) showpoint
- static const [fmtflags](#) showpos
- static const [fmtflags](#) skipws
- static const [openmode](#) trunc
- static const [fmtflags](#) unitbuf
- static const [fmtflags](#) uppercase

### Protected Types

- enum { [\\_S\\_local\\_word\\_size](#) }

### Protected Member Functions

- void [\\_M\\_cache\\_locale](#) (const [locale](#) &\_\_loc)
- void [\\_M\\_call\\_callbacks](#) ([event](#) \_\_ev) throw ()
- void [\\_M\\_dispose\\_callbacks](#) (void) throw ()
- [\\_Words](#) & [\\_M\\_grow\\_words](#) (int \_\_index, bool \_\_iword)
- void [\\_M\\_init](#) () throw ()
- template<typename [\\_ValueT](#)>  
  [\\_\\_ostream\\_type](#) & [\\_M\\_insert](#) ([\\_ValueT](#) \_\_v)
- void [init](#) ([basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \*\_\_sb)

## Protected Attributes

- `_Callback_list * _M_callbacks`
- `const __ctype_type * _M_ctype`
- `iostate _M_exception`
- `char_type _M_fill`
- `bool _M_fill_init`
- `fmtflags _M_flags`
- `locale _M_ios_locale`
- `_Words _M_local_word [_S_local_word_size]`
- `const __num_get_type * _M_num_get`
- `const __num_put_type * _M_num_put`
- `streamsize _M_precision`
- `basic_streambuf<_CharT, _Traits> * _M_streambuf`
- `iostate _M_streambuf_state`
- `basic_ostream<_CharT, _Traits> * _M_tie`
- `streamsize _M_width`
- `_Words * _M_word`
- `int _M_word_size`
- `_Words _M_word_zero`

## Friends

- class `sentry`

### 5.387.1 Detailed Description

`template<typename _CharT, typename _Traits, typename _Alloc> class std::basic_ostringstream<_CharT, _Traits, _Alloc>`

Controlling output for `std::string`.

This class supports writing to objects of type `std::basic_string`, using the inherited functions from `std::basic_ostream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

Definition at line 368 of file `sstream`.

### 5.387.2 Member Typedef Documentation

**5.387.2.1** `template<typename _CharT, typename _Traits> typedef ctype<_CharT> std::basic_ostream<_CharT, _Traits>::__ctype_type [inherited]`

These are non-standard types.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Definition at line 73 of file `ostream`.

**5.387.2.2** `template<typename _CharT, typename _Traits> typedef  
num_get<_CharT, istreambuf_iterator<_CharT, _Traits>  
> std::basic_ios<_CharT, _Traits>::__num_get_type  
[inherited]`

These are non-standard types.

Reimplemented in `std::basic_istream<_CharT, _Traits>`, `std::basic_istream<char, _Traits>`, and `std::basic_istream<char>`.

Definition at line 88 of file `basic_ios.h`.

**5.387.2.3** `template<typename _CharT, typename _Traits> typedef  
num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>  
> std::basic_ostream<_CharT, _Traits>::__num_put_type  
[inherited]`

These are non-standard types.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Definition at line 72 of file `ostream`.

**5.387.2.4** `template<typename _CharT, typename _Traits, typename _Alloc>  
typedef _CharT std::basic_ostringstream<_CharT, _Traits, _Alloc  
>::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ostream<_CharT, _Traits>`.

Definition at line 372 of file `sstream`.

**5.387.2.5** `typedef void(* std::ios_base::event_callback)(event, ios_base &, int)  
[inherited]`

The type of an event callback function.

### Parameters

- event* One of the members of the event enum.
- ios\_base* Reference to the `ios_base` object.
- int* The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 438 of file `ios_base.h`.

#### 5.387.2.6 `typedef _Ios_Fmtflags std::ios_base::fmtflags` [inherited]

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`

- basefield
- floatfield

Definition at line 257 of file ios\_base.h.

**5.387.2.7** `template<typename _CharT, typename _Traits, typename _Alloc>  
typedef traits_type::int_type std::basic_ostringstream< _CharT,  
_Traits, _Alloc >::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_ostream< \\_CharT, \\_Traits >](#).

Definition at line 377 of file sstream.

**5.387.2.8** `typedef _Ios_Iostate std::ios_base::iostate [inherited]`

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- badbit
- eofbit
- failbit
- goodbit

Definition at line 332 of file ios\_base.h.

**5.387.2.9** `template<typename _CharT, typename _Traits, typename _Alloc>  
typedef traits_type::off_type std::basic_ostringstream< _CharT,  
_Traits, _Alloc >::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_ostream< \\_CharT, \\_Traits >](#).

Definition at line 379 of file sstream.

**5.387.2.10** `typedef _Ios_Openmode std::ios_base::openmode` [inherited]

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 363 of file `ios_base.h`.

**5.387.2.11** `template<typename _CharT, typename _Traits, typename _Alloc>`  
`typedef traits_type::pos_type std::basic_ostringstream<_CharT,`  
`_Traits, _Alloc>::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ostream<_CharT, _Traits>`.

Definition at line 378 of file `sstream`.

**5.387.2.12** `typedef _Ios_Seekdir std::ios_base::seekdir` [inherited]

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 395 of file `ios_base.h`.



**5.387.2.13** `template<typename _CharT, typename _Traits, typename _Alloc>  
typedef _Traits std::basic_ostringstream<_CharT, _Traits, _Alloc  
>::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ostream<_CharT, _Traits>`.

Definition at line 373 of file `sstream`.

### 5.387.3 Member Enumeration Documentation

**5.387.3.1** `enum std::ios_base::event [inherited]`

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 421 of file `ios_base.h`.

### 5.387.4 Constructor & Destructor Documentation

**5.387.4.1** `template<typename _CharT, typename _Traits, typename  
_Alloc> std::basic_ostringstream<_CharT, _Traits, _Alloc  
>::basic_ostringstream ( ios_base::openmode __mode =  
ios_base::out ) [inline, explicit]`

Default constructor starts with an empty string buffer.

#### Parameters

*mode* Whether the buffer can read, or write, or both.

`ios_base::out` is automatically included in *mode*.

Initializes `sb` using `mode|out`, and passes `&sb` to the base class initializer. Does not allocate any buffer.

That's a lie. We initialize the base class with `NULL`, because the string class does its own memory management.

Definition at line 404 of file `sstream`.

References std::basic\_ios<\_CharT, \_Traits>::init().

**5.387.4.2** `template<typename _CharT, typename _Traits, typename  
_Alloc> std::basic_ostringstream<_CharT, _Traits, _Alloc  
>::basic_ostringstream ( const __string_type & __str,  
ios_base::openmode __mode = ios_base::out ) [inline,  
explicit]`

Starts with an existing string buffer.

#### Parameters

*str* A string to copy as a starting buffer.

*mode* Whether the buffer can read, or write, or both.

`ios_base::out` is automatically included in *mode*.

Initializes `sb` using *str* and `mode|out`, and passes `&sb` to the base class initializer.

That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

Definition at line 422 of file `sstream`.

References std::basic\_ios<\_CharT, \_Traits>::init().

**5.387.4.3** `template<typename _CharT, typename _Traits, typename  
_Alloc> std::basic_ostringstream<_CharT, _Traits, _Alloc  
>::~~basic_ostringstream ( ) [inline]`

The destructor does nothing.

The buffer is deallocated by the `stringbuf` object, not the formatting stream.

Definition at line 433 of file `sstream`.

### 5.387.5 Member Function Documentation

**5.387.5.1** `const locale& std::ios_base::_M_getloc ( ) const [inline,  
inherited]`

Locale access.

### Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 708 of file ios\_base.h.

Referenced by std::money\_get< \_CharT, \_InIter >::do\_get(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_date(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::time\_put< \_CharT, \_OutIter >::do\_put(), std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::time\_put< \_CharT, \_OutIter >::put().

**5.387.5.2** `template<typename _CharT, typename _Traits> void  
std::basic_ostream< _CharT, _Traits >::M_write ( const char_type  
* __s, streamsize __n ) [inline, inherited]`

Simple insertion.

### Parameters

*c* The character to insert.

### Returns

\*this

Tries to insert *c*.

### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 289 of file ostream.

Referenced by std::basic\_ostream< \_CharT, \_Traits >::write().

**5.387.5.3** `template<typename _CharT, typename _Traits> bool std::basic_ios<  
_CharT, _Traits >::bad ( ) const [inline, inherited]`

Fast error checking.

### Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 203 of file basic\_ios.h.

**5.387.5.4** `template<typename _CharT, typename _Traits> void  
std::basic_ios< _CharT, _Traits>::clear ( iostate __state = goodbit )  
[inherited]`

[Re]sets the error state.

### Parameters

*state* The new state flag(s) to set.

See [std::ios\\_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

Definition at line 42 of file basic\_ios.tcc.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits>::exceptions()`, `std::basic_ios< _CharT, _Traits>::rdbuf()`, and `std::basic_ios< _CharT, _Traits>::rdstate()`.

Referenced by `std::basic_ios< char, _Traits>::exceptions()`, `std::basic_fstream< _CharT, _Traits>::open()`, `std::basic_ofstream< _CharT, _Traits>::open()`, `std::basic_ifstream< _CharT, _Traits>::open()`, `std::basic_istream< _CharT, _Traits>::putback()`, `std::basic_ios< _CharT, _Traits>::rdbuf()`, `std::basic_istream< _CharT, _Traits>::seekg()`, `std::basic_ios< char, _Traits>::setstate()`, and `std::basic_istream< _CharT, _Traits>::unget()`.

**5.387.5.5** `template<typename _CharT, typename _Traits> basic_ios<  
_CharT, _Traits> & std::basic_ios< _CharT, _Traits>::copyfmt (   
const basic_ios< _CharT, _Traits> & __rhs ) [inherited]`

Copies fields of `__rhs` into this.

### Parameters

*\_\_rhs* The source values for the copies.

### Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy `exceptions()`.

Definition at line 64 of file `basic_ios.tcc`.

References `std::basic_ios< _CharT, _Traits >::exceptions()`, `std::basic_ios< _CharT, _Traits >::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios< _CharT, _Traits >::tie()`, and `std::ios_base::width()`.

#### **5.387.5.6 template<typename \_CharT, typename \_Traits> bool std::basic\_ios< \_CharT, \_Traits >::eof( ) const [inline, inherited]**

Fast error checking.

### Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 182 of file `basic_ios.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

#### **5.387.5.7 template<typename \_CharT, typename \_Traits> iostate std::basic\_ios< \_CharT, \_Traits >::exceptions( ) const [inline, inherited]**

Throwing exceptions on errors.

### Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of [exceptions\(iostate\)](#) for the meaning of the return value.

Definition at line 214 of file basic\_ios.h.

Referenced by std::basic\_ios<\_CharT, \_Traits>::clear(), and std::basic\_ios<\_CharT, \_Traits>::copyfmt().

**5.387.5.8** `template<typename _CharT, typename _Traits> void std::basic_ios<_CharT, _Traits>::exceptions ( iostate __except ) [inline, inherited]`

Throwing exceptions on errors.

#### Parameters

*except* The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type [std::ios\\_base::failure](#) is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
 std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

 std::ifstream f ("/etc/motd");

 std::cerr << "Setting badbit\n";
 f.setstate (std::ios_base::badbit);

 std::cerr << "Setting exception mask\n";
 f.exceptions (std::ios_base::badbit);
}
```

Definition at line 249 of file basic\_ios.h.

**5.387.5.9** `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::fail ( ) const [inline, inherited]`

Fast error checking.

### Returns

True if either the badbit or the failbit is set.

Checking the badbit in [fail\(\)](#) is historical practice. Note that other iostate flags may also be set.

Definition at line 193 of file basic\_ios.h.

Referenced by `std::basic_ios< char, _Traits >::operator void *()`, `std::basic_ios< char, _Traits >::operator!()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::regex_traits< _Ch_type >::value()`.

**5.387.5.10** `template<typename _CharT, typename _Traits> char_type  
std::basic_ios< _CharT, _Traits >::fill ( ) const [inline,  
inherited]`

Retrieves the *empty* character.

### Returns

The current fill character.

It defaults to a space ( ' ' ) in the current locale.

Definition at line 362 of file basic\_ios.h.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, and `std::basic_ios< char, _Traits >::fill()`.

**5.387.5.11** `template<typename _CharT, typename _Traits> char_type  
std::basic_ios< _CharT, _Traits >::fill ( char_type __ch )  
[inline, inherited]`

Sets a new *empty* character.

### Parameters

*ch* The new character.

### Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and  $Q < P$ . It defaults to a space ( ' ') in the current locale.

Definition at line 382 of file `basic_ios.h`.

#### **5.387.5.12 fmtflags std::ios\_base::flags ( ) const [inline, inherited]**

Access to format flags.

##### **Returns**

The format control flags for both input and output.

Definition at line 553 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, `std::operator>>()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

#### **5.387.5.13 fmtflags std::ios\_base::flags ( fmtflags \_\_fmtfl ) [inline, inherited]**

Setting new format flags all at once.

##### **Parameters**

*\_\_fmtfl* The new flags to set.

##### **Returns**

The previous format control flags.

This function overwrites all the format flags with *\_\_fmtfl*.

Definition at line 564 of file `ios_base.h`.

#### **5.387.5.14 template<typename \_CharT, typename \_Traits > basic\_ostream< \_CharT, \_Traits > & std::basic\_ostream< \_CharT, \_Traits >::flush ( ) [inherited]**

Synchronizing the stream buffer.



### Returns

\*this

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf() ->pubsync()`, and if that returns -1, sets badbit.

Definition at line 213 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

Referenced by `std::flush()`.

### 5.387.5.15 locale std::ios\_base::getloc( ) const [inline, inherited]

Locale access.

### Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 697 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::money_put< _CharT, _OutIter >::do_put()`, `std::basic_ios< _CharT, _Traits >::imbue()`, `std::operator>>()`, and `std::ws()`.

### 5.387.5.16 template<typename \_CharT, typename \_Traits> bool std::basic\_ios< \_CharT, \_Traits >::good( ) const [inline, inherited]

Fast error checking.

### Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 172 of file `basic_ios.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

**5.387.5.17** `template<typename _CharT, typename _Traits > locale  
std::basic_ios< _CharT, _Traits >::imbue ( const locale & __loc )  
[inherited]`

Moves to a new locale.

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional 110n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Reimplemented from `std::ios_base`.

Definition at line 115 of file `basic_ios.tcc`.

References `std::ios_base::getloc()`, and `std::basic_ios< _CharT, _Traits >::rdbuf()`.

Referenced by `std::operator<<()`.

**5.387.5.18** `template<typename _CharT, typename _Traits> void  
std::basic_ios< _CharT, _Traits >::init ( basic_streambuf<  
_CharT, _Traits > * __sb ) [protected, inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 127 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

Referenced by `std::basic_fstream< _CharT, _Traits >::basic_fstream()`, `std::basic_ifstream< _CharT, _Traits >::basic_ifstream()`, `std::basic_ios< char, _Traits >::basic_ios()`, `std::basic_istream< char >::basic_istream()`, `std::basic_istreamstream< _CharT, _Traits, _Alloc >::basic_istreamstream()`, `std::basic_ofstream< _CharT, _Traits >::basic_ofstream()`, `std::basic_ostream< char >::basic_ostream()`, `std::basic_ostringstream< _CharT, _Traits, _Alloc >::basic_ostringstream()`, and `std::basic_stringstream< _CharT, _Traits, _Alloc >::basic_stringstream()`.

**5.387.5.19** long& std::ios\_base::iword ( int \_\_ix ) [inline, inherited]

Access to integer array.

**Parameters**

*\_\_ix* Index into the array.

**Returns**

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 743 of file ios\_base.h.

**5.387.5.20** template<typename \_CharT, typename \_Traits> char  
std::basic\_ios<\_CharT, \_Traits>::narrow ( char\_type \_\_c, char  
\_\_dfault ) const [inline, inherited]

Squeezes characters.

**Parameters**

*c* The character to narrow.

*dfault* The character to narrow.

**Returns**

The narrowed character.

Maps a character of char\_type to a character of char, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, dfault)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.ht>

Definition at line 422 of file basic\_ios.h.

**5.387.5.21** `template<typename _CharT, typename _Traits> std::basic_ios<  
_CharT, _Traits >::operator void * ( ) const [inline,  
inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 113 of file `basic_ios.h`.

**5.387.5.22** `template<typename _CharT, typename _Traits> bool  
std::basic_ios<_CharT, _Traits >::operator! ( ) const  
[inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 117 of file `basic_ios.h`.

**5.387.5.23** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream<_CharT, _Traits >::operator<< ( long __n )  
[inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 167 of file `ostream`.

**5.387.5.24** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream<_CharT, _Traits>::operator<<( double __f  
) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 211 of file `ostream`.

**5.387.5.25** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream<_CharT, _Traits>::operator<<(   
__ostream_type &(*)(__ostream_type &) __pf ) [inline,  
inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "`std::cout << std::endl`". For more information, see the `iomanip` header.

Definition at line 110 of file `ostream`.

**5.387.5.26** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream<_CharT, _Traits>::operator<<( long double  
__f ) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

*\*this* if successful

## 5.387 `std::basic_ostringstream<_CharT, _Traits, _Alloc>` Class Template Reference 2131

---

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 223 of file `ostream`.

**5.387.5.27** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< ( bool __n ) [inline, inherited]`

Basic arithmetic inserters.

### Parameters

*A* variable of builtin type.

### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 175 of file `ostream`.

**5.387.5.28** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< ( const void * __p ) [inline, inherited]`

Basic arithmetic inserters.

### Parameters

*A* variable of builtin type.

### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 227 of file `ostream`.

**5.387.5.29** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream<_CharT, _Traits>::operator<<( __ios_type  
&(*)(__ios_type &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the `iomanip` header.

Definition at line 119 of file `ostream`.

**5.387.5.30** `template<typename _CharT, typename _Traits> basic_ostream<  
_CharT, _Traits> & std::basic_ostream<_CharT, _Traits  
>::operator<<( __streambuf_type * __sb ) [inherited]`

Extracting from another streambuf.

#### Parameters

*sb* A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from *sb* and inserted into *\*this* until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from *sb*, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 122 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.387.5.31** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< ( long long __n ) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 202 of file ostream.

**5.387.5.32** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< ( unsigned short __n ) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 182 of file ostream.

**5.387.5.33** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<< ( int __n ) [inherited]`

Basic arithmetic inserters.



### Parameters

*A* variable of builtin type.

### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 108 of file `ostream.tcc`.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

**5.387.5.34** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( ios_base &(*) (ios_base &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

Definition at line 129 of file `ostream`.

**5.387.5.35** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( unsigned int __n ) [inline, inherited]`

Basic arithmetic inserters.

### Parameters

*A* variable of builtin type.

### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 193 of file `ostream`.

**5.387.5.36** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream<_CharT, _Traits>::operator<<( unsigned  
long __n ) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 171 of file `ostream`.

**5.387.5.37** `template<typename _CharT, typename _Traits> basic_ostream<  
_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>  
>::operator<<( short __n ) [inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 94 of file `ostream.tcc`.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

**5.387.5.38** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream<_CharT, _Traits>::operator<<( float __f )  
[inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 215 of file ostream.

**5.387.5.39** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream<_CharT, _Traits>::operator<< ( unsigned  
long long __n ) [inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 206 of file ostream.

**5.387.5.40** `streamsize std::ios_base::precision ( streamsize __prec )  
[inline, inherited]`

Changing flags.

**Parameters**

*prec* The new precision value.

**Returns**

The previous value of `precision()`.

Definition at line 632 of file ios\_base.h.

**5.387.5.41** `streamsize std::ios_base::precision ( ) const [inline, inherited]`

Flags access.

#### Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 623 of file ios\_base.h.

Referenced by std::basic\_ios< \_CharT, \_Traits >::copyfmt(), and std::operator<<().

**5.387.5.42** `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put ( char_type __c ) [inherited]`

Simple insertion.

#### Parameters

*c* The character to insert.

#### Returns

\*this

Tries to insert *c*.

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 151 of file ostream.tcc.

References std::ios\_base::badbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), and std::basic\_ios< \_CharT, \_Traits >::setstate().

Referenced by std::endl(), and std::ends().

**5.387.5.43** `void*& std::ios_base::pword ( int __ix ) [inline, inherited]`

Access to void pointer array.

#### Parameters

`__ix` Index into the array.

#### Returns

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 764 of file `ios_base.h`.

**5.387.5.44** `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rdbuf ( basic_streambuf<_CharT, _Traits> * __sb ) [inherited]`

Changing the underlying buffer.

#### Parameters

`sb` The new stream buffer.

#### Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream foo; // or some other derived type
std::streambuf* p =;

foo.ios::rdbuf(p); // ios == basic_ios<char>
```

Definition at line 54 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::clear()`.

**5.387.5.45** `template<typename _CharT, typename _Traits, typename _Alloc>  
__stringbuf_type* std::basic_ostringstream<_CharT, _Traits,  
_Alloc>::rdbuf ( ) const [inline]`

Accessing the underlying buffer.

#### Returns

The current [basic\\_stringbuf](#) buffer.

This hides both signatures of [std::basic\\_ios::rdbuf\(\)](#).

Reimplemented from [std::basic\\_ios<\\_CharT, \\_Traits>](#).

Definition at line 444 of file `sstream`.

**5.387.5.46** `template<typename _CharT, typename _Traits> iostate  
std::basic_ios<_CharT, _Traits>::rdstate ( ) const [inline,  
inherited]`

Returns the error state of the stream buffer.

#### Returns

A bit pattern (well, isn't everything?)

See [std::ios\\_base::iostate](#) for the possible bit values. Most users will call one of the interpreting wrappers, e.g., [good\(\)](#).

Definition at line 129 of file `basic_ios.h`.

Referenced by `std::basic_ios<char, _Traits>::bad()`, `std::basic_ios<_CharT, _Traits>::clear()`, `std::basic_ios<char, _Traits>::eof()`, `std::basic_ios<char, _Traits>::fail()`, `std::basic_ios<char, _Traits>::good()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ios<char, _Traits>::setstate()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

**5.387.5.47** `void std::ios_base::register_callback ( event_callback __fn, int  
__index ) [inherited]`

Add the callback `__fn` with parameter `__index`.

#### Parameters

- \_\_fn* The function to add.
- \_\_index* The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**5.387.5.48** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp( pos_type __pos ) [inherited]`

Changing the current write position.

#### Parameters

- pos* A file position object.

#### Returns

\*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.

Definition at line 260 of file ostream.tcc.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.387.5.49** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp( off_type __off, ios_base::seekdir __dir ) [inherited]`

Changing the current write position.

#### Parameters

- off* A file offset object.
- dir* The direction in which to seek.

### Returns

\*this

If `fail()` is not true, calls `rddbuf() -> pubseekoff(off, dir)`. If that function fails, sets failbit.

Definition at line 292 of file ostream.tcc.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios<_CharT, _Traits>::rddbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

#### 5.387.5.50 `fmtflags std::ios_base::setf ( fmtflags __fmtfl, fmtflags __mask )` `[inline, inherited]`

Setting new format flags.

### Parameters

*fmtfl* Additional flags to set.

*mask* The flags mask for *fmtfl*.

### Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 597 of file ios\_base.h.

#### 5.387.5.51 `fmtflags std::ios_base::setf ( fmtflags __fmtfl )` `[inline, inherited]`

Setting new format flags.

### Parameters

*fmtfl* Additional flags to set.

### Returns

The previous format control flags.



This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 580 of file ios\_base.h.

Referenced by `std::dec()`, `std::fixed()`, `std::hex()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

**5.387.5.52** `template<typename _CharT, typename _Traits> void  
std::basic_ios< _CharT, _Traits >::setstate ( iostate __state )  
[inline, inherited]`

Sets additional flags in the error state.

#### Parameters

*state* The additional state flag(s) to set.

See [std::ios\\_base::iostate](#) for the possible bit values.

Definition at line 149 of file basic\_ios.h.

Referenced by `std::basic_ostream< char >::_M_write()`, `std::basic_fstream< _CharT, _Traits >::close()`, `std::basic_ofstream< _CharT, _Traits >::close()`, `std::basic_ifstream< _CharT, _Traits >::close()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_fstream< _CharT, _Traits >::open()`, `std::basic_ofstream< _CharT, _Traits >::open()`, `std::basic_ifstream< _CharT, _Traits >::open()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

**5.387.5.53** `template<typename _CharT, typename _Traits, typename _Alloc>  
__string_type std::basic_ostringstream< _CharT, _Traits, _Alloc  
>::str ( ) const [inline]`

Copying out the string buffer.

**Returns**

`rdbuf()` -> `str()`

Definition at line 452 of file sstream.

Referenced by `std::operator<<()`.

**5.387.5.54** `template<typename _CharT, typename _Traits, typename _Alloc>  
void std::basic_ostringstream<_CharT, _Traits, _Alloc>::str (  
const __string_type & __s ) [inline]`

Setting a new buffer.

**Parameters**

*s* The string to use as a new sequence.

Calls `rdbuf()` -> `str(s)`.

Definition at line 462 of file sstream.

**5.387.5.55** `static bool std::ios_base::sync_with_stdio ( bool __sync = true )  
[static, inherited]`

Interaction with the standard C I/O objects.

**Parameters**

*sync* Whether to synchronize or not.

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

**5.387.5.56** `template<typename _CharT, typename _Traits> basic_ostream<  
_CharT, _Traits>::pos_type std::basic_ostream<_CharT, _Traits  
>::tellp ( ) [inherited]`

Getting the current write position.

#### Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rddbuf()->pubseekoff(0, cur, out)`.

Definition at line 239 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::out`, and `std::basic_ios<_CharT, _Traits>::rddbuf()`.

**5.387.5.57** `template<typename _CharT, typename _Traits>  
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits  
>::tie ( ) const [inline, inherited]`

Fetches the current *tied* stream.

#### Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 287 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

**5.387.5.58** `template<typename _CharT, typename _Traits>  
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits  
>::tie ( basic_ostream<_CharT, _Traits> * __tiestr ) [inline,  
inherited]`

Ties this stream to an output stream.

#### Parameters

*tiestr* The output stream.

#### Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see [tie\(\)](#) for more.

Definition at line 299 of file `basic_ios.h`.

**5.387.5.59** `void std::ios_base::unsetf ( fmtflags __mask ) [inline, inherited]`

Clearing format flags.

#### Parameters

*mask* The flags to unset.

This function clears *mask* in the format flags.

Definition at line 612 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

**5.387.5.60** `template<typename _CharT, typename _Traits> char_type  
std::basic_ios<_CharT, _Traits>::widen ( char __c ) const  
[inline, inherited]`

Widens characters.

#### Parameters

*c* The character to widen.

#### Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

## 5.387 std::basic\_ostringstream<\_CharT, \_Traits, \_Alloc> Class Template Reference 2146

---

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 441 of file basic\_ios.h.

Referenced by std::endl(), std::basic\_ios<char, \_Traits>::fill(), std::basic\_istream<char>::get(), std::basic\_istream<char>::getline(), std::getline(), and std::operator>>().

### 5.387.5.61 streamsize std::ios\_base::width ( streamsize \_\_wide ) [inline, inherited]

Changing flags.

#### Parameters

*wide* The new width value.

#### Returns

The previous value of [width\(\)](#).

Definition at line 655 of file ios\_base.h.

### 5.387.5.62 streamsize std::ios\_base::width ( ) const [inline, inherited]

Flags access.

#### Returns

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 646 of file ios\_base.h.

Referenced by std::basic\_ios<\_CharT, \_Traits>::copyfmt(), std::num\_put<\_CharT, \_OutIter>::do\_put(), and std::operator>>().

### 5.387.5.63 template<typename \_CharT, typename \_Traits> basic\_ostream<\_CharT, \_Traits> & std::basic\_ostream<\_CharT, \_Traits>::write ( const char\_type \* \_\_s, streamsize \_\_n ) [inherited]

Character string insertion.

#### Parameters

- s* The array to insert.
- n* Maximum number of characters to insert.

#### Returns

\*this

Characters are copied from *s* and inserted into the stream until one of the following happens:

- *n* characters are inserted
- inserting into the output sequence fails (in this case, badbit will be set in the stream's error state)

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 185 of file ostream.tcc.

References `std::basic_ostream< _CharT, _Traits >::_M_write()`, and `std::ios_base::badbit`.

#### 5.387.5.64 `static int std::ios_base::xalloc ( ) throw () [static, inherited]`

Access to unique indices.

#### Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

## **5.387.6 Member Data Documentation**

### **5.387.6.1 const fmtflags std::ios\_base::adjustfield [static, inherited]**

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 312 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

### **5.387.6.2 const openmode std::ios\_base::app [static, inherited]**

Seek to end before each write.

Definition at line 366 of file `ios_base.h`.

### **5.387.6.3 const openmode std::ios\_base::ate [static, inherited]**

Open and seek to end immediately after opening.

Definition at line 369 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

### **5.387.6.4 const iostate std::ios\_base::badbit [static, inherited]**

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 336 of file `ios_base.h`.

Referenced by `std::basic_ostream< char >::_M_write()`, `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::basic_ios< char, _Traits >::fail()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ios< _CharT, _Traits >::init()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _`

## 5.387 `std::basic_ostringstream<_CharT, _Traits, _Alloc>` Class Template Reference 2149

---

`Traits >::read()`, `std::basic_istream<_CharT, _Traits >::readsome()`, `std::basic_istream<_CharT, _Traits >::seekg()`, `std::basic_ostream<_CharT, _Traits >::seekp()`, `std::basic_istream<_CharT, _Traits >::sync()`, `std::basic_istream<_CharT, _Traits >::tellg()`, `std::basic_ostream<_CharT, _Traits >::tellp()`, `std::basic_istream<_CharT, _Traits >::unget()`, `std::basic_ostream<_CharT, _Traits >::write()`, and `std::basic_ostream<_CharT, _Traits >::sentry::~~sentry()`.

### 5.387.6.5 `const fmtflags std::ios_base::basefield` [`static`, `inherited`]

A mask of `dec|oct|hex`. Useful for the 2-arg form of `setf`.

Definition at line 315 of file `ios_base.h`.

Referenced by `std::dec()`, `std::num_get<_CharT, _InIter >::do_get()`, `std::hex()`, `std::oct()`, and `std::basic_ostream<_CharT, _Traits >::operator<<()`.

### 5.387.6.6 `const seekdir std::ios_base::beg` [`static`, `inherited`]

Request a seek relative to the beginning of the stream.

Definition at line 398 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits >::seekpos()`.

### 5.387.6.7 `const openmode std::ios_base::binary` [`static`, `inherited`]

Perform input and output in binary mode (as opposed to text mode). `/// This is probably not what you think it is; see http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html.`

Definition at line 374 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits >::showmanyc()`.

### 5.387.6.8 `const fmtflags std::ios_base::boolalpha` [`static`, `inherited`]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 260 of file `ios_base.h`.



## **5.387 std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc > Class Template Reference** **2150**

---

Referenced by `std::boolalpha()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::noboolalpha()`.

### **5.387.6.9 const seekdir std::ios\_base::cur [static, inherited]**

Request a seek relative to the current position within the sequence.

Definition at line 401 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_filebuf< _CharT, _Traits >::seekoff()`, `std::basic_istream< _CharT, _Traits >::tellg()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

### **5.387.6.10 const fmtflags std::ios\_base::dec [static, inherited]**

Converts integer input or generates integer output in decimal base.

Definition at line 263 of file `ios_base.h`.

Referenced by `std::dec()`.

### **5.387.6.11 const seekdir std::ios\_base::end [static, inherited]**

Request a seek relative to the current end of the sequence.

Definition at line 404 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`.

### **5.387.6.12 const iostate std::ios\_base::eofbit [static, inherited]**

Indicates that an input operation reached the end of an input sequence.

Definition at line 339 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`,

std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::eof(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::unget(), and std::ws().

**5.387.6.13 const iostate std::ios\_base::failbit [static, inherited]**

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 344 of file ios\_base.h.

Referenced by std::basic\_fstream< \_CharT, \_Traits >::close(), std::basic\_ofstream< \_CharT, \_Traits >::close(), std::basic\_ifstream< \_CharT, \_Traits >::close(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::fail(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_fstream< \_CharT, \_Traits >::open(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry(), and std::basic\_istream< \_CharT, \_Traits >::sentry::sentry().

**5.387.6.14 const fmtflags std::ios\_base::fixed [static, inherited]**

Generate floating-point output in fixed-point notation.

Definition at line 266 of file ios\_base.h.

Referenced by std::fixed().

**5.387.6.15 const fmtflags std::ios\_base::floatfield [static, inherited]**

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 318 of file `ios_base.h`.

Referenced by `std::fixed()`, and `std::scientific()`.

#### **5.387.6.16 const iostate std::ios\_base::goodbit [static, inherited]**

Indicates all is well.

Definition at line 347 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ios< _CharT, _Traits >::init()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::basic_istream< _CharT, _Traits >::sync()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

#### **5.387.6.17 const fmtflags std::ios\_base::hex [static, inherited]**

Converts integer input or generates integer output in hexadecimal base.

Definition at line 269 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::hex()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

#### **5.387.6.18 const openmode std::ios\_base::in [static, inherited]**

Open for input. Default for `ifstream` and `fstream`.

Definition at line 377 of file `ios_base.h`.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_set_buffer()`, `std::basic_ifstream< _CharT, _Traits >::open()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

#### **5.387.6.19 const fmtflags std::ios\_base::internal [static, inherited]**

Adds fill characters at a designated internal point in certain `///` generated output, or identical to `right` if no such point is `///` designated.

Definition at line 274 of file `ios_base.h`.

Referenced by `std::internal()`.

#### **5.387.6.20 const fmtflags std::ios\_base::left [static, inherited]**

Adds fill characters on the right (final positions) of certain `///` generated output. (I.e., the thing you print is flush left.).

Definition at line 278 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, and `std::left()`.

#### **5.387.6.21 const fmtflags std::ios\_base::oct [static, inherited]**

Converts integer input or generates integer output in octal base.

Definition at line 281 of file `ios_base.h`.

Referenced by `std::oct()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

#### **5.387.6.22 const openmode std::ios\_base::out [static, inherited]**

Open for output. Default for `ofstream` and `fstream`.

Definition at line 380 of file `ios_base.h`.

### 5.387 `std::basic_ostringstream< _CharT, _Traits, _Alloc >` Class Template Reference 2154

---

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_set_buffer()`, `std::basic_ofstream< _CharT, _Traits >::open()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

#### 5.387.6.23 `const fmtflags std::ios_base::right` **[static, inherited]**

Adds fill characters on the left (initial positions) of certain `///` generated output. (I.e., the thing you print is flush right.).

Definition at line 285 of file `ios_base.h`.

Referenced by `std::right()`.

#### 5.387.6.24 `const fmtflags std::ios_base::scientific` **[static, inherited]**

Generates floating-point output in scientific notation.

Definition at line 288 of file `ios_base.h`.

Referenced by `std::scientific()`.

#### 5.387.6.25 `const fmtflags std::ios_base::showbase` **[static, inherited]**

Generates a prefix indicating the numeric base of generated integer `///` output.

Definition at line 292 of file `ios_base.h`.

Referenced by `std::noshowbase()`, and `std::showbase()`.

#### 5.387.6.26 `const fmtflags std::ios_base::showpoint` **[static, inherited]**

Generates a decimal-point character unconditionally in generated `///` floating-point output.

Definition at line 296 of file `ios_base.h`.

Referenced by `std::noshowpoint()`, and `std::showpoint()`.

**5.387.6.27 const fmtflags std::ios\_base::showpos [static, inherited]**

Generates a + sign in non-negative generated numeric output.

Definition at line 299 of file ios\_base.h.

Referenced by std::noshowpos(), and std::showpos().

**5.387.6.28 const fmtflags std::ios\_base::skipws [static, inherited]**

Skips leading white space before certain input operations.

Definition at line 302 of file ios\_base.h.

Referenced by std::noskipws(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), and std::skipws().

**5.387.6.29 const openmode std::ios\_base::trunc [static, inherited]**

Open for input. Default for ostream.

Definition at line 383 of file ios\_base.h.

**5.387.6.30 const fmtflags std::ios\_base::unitbuf [static, inherited]**

Flushes output after each output operation.

Definition at line 305 of file ios\_base.h.

Referenced by std::nunitbuf(), std::unitbuf(), and std::basic\_ostream< \_CharT, \_Traits >::sentry::~sentry().

**5.387.6.31 const fmtflags std::ios\_base::uppercase [static, inherited]**

Replaces certain lowercase letters with their uppercase equivalents /// in generated output.

Definition at line 309 of file ios\_base.h.

## 5.388 `std::basic_regex< _Ch_type, _Rx_traits >` Class Template Reference 2156

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, `std::nouppercase()`, and `std::uppercase()`.

The documentation for this class was generated from the following file:

- [sstream](#)

## 5.388 `std::basic_regex< _Ch_type, _Rx_traits >` Class Template Reference

### Public Types

- typedef [regex\\_constants::syntax\\_option\\_type](#) **flag\_type**
- typedef `_Rx_traits::locale_type` **locale\_type**
- typedef `_Rx_traits::string_type` **string\_type**
- typedef `_Ch_type` **value\_type**

### Public Member Functions

- [basic\\_regex](#) ()
- [basic\\_regex](#) (const `_Ch_type` \*\_\_p, `flag_type` \_\_f=[regex\\_constants::ECMAScript](#))
- [basic\\_regex](#) (const [basic\\_regex](#) &\_\_rhs)
- [basic\\_regex](#) ([initializer\\_list](#)< `_Ch_type` > \_\_l, `flag_type` \_\_f=[regex\\_constants::ECMAScript](#))
- [basic\\_regex](#) (const [basic\\_regex](#) &&\_\_rhs)
- [basic\\_regex](#) (const `_Ch_type` \*\_\_p, `std::size_t` \_\_len, `flag_type` \_\_f)
- [template](#)<typename `_Ch_traits`, typename `_Ch_alloc` >  
[basic\\_regex](#) (const [std::basic\\_string](#)< `_Ch_type`, `_Ch_traits`, `_Ch_alloc` > &\_\_s, `flag_type` \_\_f=[regex\\_constants::ECMAScript](#))
- [template](#)<typename `_InputIterator` >  
[basic\\_regex](#) (`_InputIterator` \_\_first, `_InputIterator` \_\_last, `flag_type` \_\_f=[regex\\_constants::ECMAScript](#))
- [~basic\\_regex](#) ()
- const `__regex::_AutomatonPtr` & [\\_M\\_get\\_automaton](#) () const
- [template](#)<typename `_InputIterator` >  
[basic\\_regex](#) & [assign](#) (`_InputIterator` \_\_first, `_InputIterator` \_\_last, `flag_type` \_\_flags=[regex\\_constants::ECMAScript](#))
- [basic\\_regex](#) & [assign](#) (const `_Ch_type` \*\_\_p, `flag_type` \_\_flags=[regex\\_constants::ECMAScript](#))
- [basic\\_regex](#) & [assign](#) (const `_Ch_type` \*\_\_p, `std::size_t` \_\_len, `flag_type` \_\_flags)
- [template](#)<typename `_Ch_traits`, typename `_Allocator` >  
[basic\\_regex](#) & [assign](#) (const [basic\\_string](#)< `_Ch_type`, `_Ch_traits`, `_Allocator` > &\_\_s, `flag_type` \_\_f=[regex\\_constants::ECMAScript](#))

- `basic_regex` & `assign` (`initializer_list< _Ch_type > __l`, `flag_type __f=regex_constants::ECMAScript`)
- `basic_regex` & `assign` (`const basic_regex &__rhs`)
- `basic_regex` & `assign` (`basic_regex &&__rhs`)
- `flag_type flags` () `const`
- `locale_type getloc` () `const`
- `locale_type imbue` (`locale_type __loc`)
- `unsigned int mark_count` () `const`
- `basic_regex` & `operator=` (`const _Ch_type *__p`)
- `basic_regex` & `operator=` (`const basic_regex &__rhs`)
- `basic_regex` & `operator=` (`basic_regex &&__rhs`)
- `template<typename _Ch_type_traits, typename _Allocator >`  
`basic_regex` & `operator=` (`const basic_string< _Ch_type, _Ch_type_traits, _Allocator > &__s`)
- `void swap` (`basic_regex &__rhs`)

## Static Public Attributes

### Constants

*std [28.8.1](1)*

- static constexpr `regex_constants::syntax_option_type` `icase`
- static constexpr `regex_constants::syntax_option_type` `nosubs`
- static constexpr `regex_constants::syntax_option_type` `optimize`
- static constexpr `regex_constants::syntax_option_type` `collate`
- static constexpr `regex_constants::syntax_option_type` `ECMAScript`
- static constexpr `regex_constants::syntax_option_type` `basic`
- static constexpr `regex_constants::syntax_option_type` `extended`
- static constexpr `regex_constants::syntax_option_type` `awk`
- static constexpr `regex_constants::syntax_option_type` `grep`
- static constexpr `regex_constants::syntax_option_type` `egrep`

## Protected Attributes

- `__regex::AutomatonPtr` `_M_automaton`
- `flag_type` `_M_flags`
- `_Rx_traits` `_M_traits`

### 5.388.1 Detailed Description

`template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> class std::basic_regex< _Ch_type, _Rx_traits >`

Objects of specializations of this class represent regular expressions constructed from sequences of character type `_Ch_type`.



## 5.388 `std::basic_regex<_Ch_type, _Rx_traits>` Class Template Reference 2158

---

Storage for the regular expression is allocated and deallocated as necessary by the member functions of this class.

Definition at line 341 of file `regex.h`.

### 5.388.2 Constructor & Destructor Documentation

**5.388.2.1** `template<typename _Ch_type, typename _Rx_traits =  
regex_traits<_Ch_type>> std::basic_regex<_Ch_type, _Rx_traits  
>::basic_regex ( ) [inline]`

Constructs a basic regular expression that does not match any character sequence.

Definition at line 382 of file `regex.h`.

**5.388.2.2** `template<typename _Ch_type, typename _Rx_traits =  
regex_traits<_Ch_type>> std::basic_regex<_Ch_type, _Rx_traits  
>::basic_regex ( const _Ch_type * __p, flag_type __f =  
regex_constants::ECMAScript ) [inline, explicit]`

Constructs a basic regular expression from the sequence `[p, p + char_traits<_Ch_type>::length(p))` interpreted according to the flags in `f`.

#### Parameters

*p* A pointer to the start of a C-style null-terminated string containing a regular expression.

*f* Flags indicating the syntax rules and options.

#### Exceptions

[\*regex\\_error\*](#) if `p` is not a valid regular expression.

Definition at line 400 of file `regex.h`.

**5.388.2.3** `template<typename _Ch_type, typename _Rx_traits =  
regex_traits<_Ch_type>> std::basic_regex<_Ch_type, _Rx_traits  
>::basic_regex ( const _Ch_type * __p, std::size_t __len, flag_type  
__f ) [inline]`

Constructs a basic regular expression from the sequence `[p, p + len)` interpreted according to the flags in `f`.

### Parameters

- p* A pointer to the start of a string containing a regular expression.
- len* The length of the string containing the regular expression.
- f* Flags indicating the syntax rules and options.

### Exceptions

- regex\_error* if *p* is not a valid regular expression.

Definition at line 418 of file `regex.h`.

**5.388.2.4** `template<typename _Ch_type, typename _Rx_traits =  
regex_traits<_Ch_type>> std::basic_regex< _Ch_type, _Rx_traits  
>::basic_regex ( const basic_regex< _Ch_type, _Rx_traits > &  
__rhs ) [inline]`

Copy-constructs a basic regular expression.

### Parameters

- rhs* A `regex` object.

Definition at line 428 of file `regex.h`.

**5.388.2.5** `template<typename _Ch_type, typename _Rx_traits =  
regex_traits<_Ch_type>> std::basic_regex< _Ch_type, _Rx_traits  
>::basic_regex ( const basic_regex< _Ch_type, _Rx_traits > &&  
__rhs ) [inline]`

Move-constructs a basic regular expression.

### Parameters

- rhs* A `regex` object.

Definition at line 438 of file `regex.h`.

**5.388.2.6** `template<typename _Ch_type, typename _Rx_traits =  
regex_traits<_Ch_type>> template<typename _Ch_traits ,  
typename _Ch_alloc > std::basic_regex< _Ch_type, _Rx_traits  
>::basic_regex ( const std::basic_string< _Ch_type, _Ch_traits,  
_Ch_alloc > & __s, flag_type __f = regex_constants::ECMAScript )  
[inline, explicit]`

Constructs a basic regular expression from the string *s* interpreted according to the flags in *f*.

#### Parameters

- s* A string containing a regular expression.
- f* Flags indicating the syntax rules and options.

#### Exceptions

*regex\_error* if *s* is not a valid regular expression.

Definition at line 454 of file regex.h.

**5.388.2.7** `template<typename _Ch_type, typename _Rx_traits =  
regex_traits<_Ch_type>> template<typename _InputIterator  
> std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (   
_InputIterator __first, _InputIterator __last, flag_type __f =  
regex_constants::ECMAScript ) [inline]`

Constructs a basic regular expression from the range [*first*, *last*) interpreted according to the flags in *f*.

#### Parameters

- first* The start of a range containing a valid regular expression.
- last* The end of a range containing a valid regular expression.
- f* The format flags of the regular expression.

#### Exceptions

*regex\_error* if [*first*, *last*) is not a valid regular expression.

Definition at line 476 of file regex.h.

## 5.388 `std::basic_regex<_Ch_type, _Rx_traits>` Class Template Reference 2161

---

**5.388.2.8** `template<typename _Ch_type, typename _Rx_traits =  
regex_traits<_Ch_type>> std::basic_regex<_Ch_type, _Rx_traits  
>::basic_regex ( initializer_list<_Ch_type> __l, flag_type __f =  
regex_constants::ECMAScript ) [inline]`

Constructs a basic regular expression from an initializer list.

### Parameters

- l* The initializer list.
- f* The format flags of the regular expression.

### Exceptions

*regex\_error* if *l* is not a valid regular expression.

Definition at line 490 of file `regex.h`.

**5.388.2.9** `template<typename _Ch_type, typename _Rx_traits =  
regex_traits<_Ch_type>> std::basic_regex<_Ch_type, _Rx_traits  
>::~~basic_regex ( ) [inline]`

Destroys a basic regular expression.

Definition at line 500 of file `regex.h`.

## 5.388.3 Member Function Documentation

**5.388.3.1** `template<typename _Ch_type, typename _Rx_traits =  
regex_traits<_Ch_type>> basic_regex& std::basic_regex<  
_Ch_type, _Rx_traits>::assign ( const basic_regex<_Ch_type,  
_Rx_traits> & __rhs ) [inline]`

the real assignment operator.

### Parameters

- rhs* Another regular expression object.

Definition at line 546 of file `regex.h`.

## 5.388 `std::basic_regex<_Ch_type, _Rx_traits>` Class Template Reference 2162

References `std::basic_regex<_Ch_type, _Rx_traits>::swap()`.

Referenced by `std::basic_regex<_Ch_type, _Rx_traits>::operator=()`.

**5.388.3.2** `template<typename _Ch_type, typename _Rx_traits =  
regex_traits<_Ch_type>> basic_regex& std::basic_regex<  
_Ch_type, _Rx_traits>::assign ( const _Ch_type * __p, std::size_t  
__len, flag_type __flags ) [inline]`

Assigns a new regular expression to a regex object from a C-style string containing a regular expression pattern.

### Parameters

*p* A pointer to a C-style string containing a regular expression pattern.

*len* The length of the regular expression pattern string.

*flags* Syntax option flags.

### Exceptions

*regex\_error* if *p* does not contain a valid regular expression pattern interpreted according to *flags*. If *regex\_error* is thrown, *\*this* remains unchanged.

Definition at line 598 of file `regex.h`.

References `std::basic_regex<_Ch_type, _Rx_traits>::assign()`.

Referenced by `std::basic_regex<_Ch_type, _Rx_traits>::assign()`.

**5.388.3.3** `template<typename _Ch_type, typename _Rx_traits =  
regex_traits<_Ch_type>> template<typename _Ch_type_traits ,  
typename _Allocator > basic_regex& std::basic_regex<_Ch_type,  
_Rx_traits>::assign ( const basic_string<_Ch_type, _Ch_type_traits,  
_Allocator > & __s, flag_type __f = regex_constants::ECMAScript  
) [inline]`

Assigns a new regular expression to a regex object from a string containing a regular expression pattern.

### Parameters

*s* A string containing a regular expression pattern.

*flags* Syntax option flags.

## 5.388 `std::basic_regex<_Ch_type, _Rx_traits>` Class Template Reference 2163

### Exceptions

**`regex_error`** if `p` does not contain a valid regular expression pattern interpreted according to `flags`. If **`regex_error`** is thrown, `*this` remains unchanged.

Definition at line 614 of file `regex.h`.

References `std::basic_regex<_Ch_type, _Rx_traits>::swap()`.

```
5.388.3.4 template<typename _Ch_type, typename _Rx_traits =
 regex_traits<_Ch_type>> template<typename _InputIterator >
 basic_regex& std::basic_regex<_Ch_type, _Rx_traits>::assign (
 _InputIterator __first, _InputIterator __last, flag_type __flags =
 regex_constants::ECMAScript) [inline]
```

Assigns a new regular expression to a regex object.

### Parameters

*first* The start of a range containing a valid regular expression.

*last* The end of a range containing a valid regular expression.

*flags* Syntax option flags.

### Exceptions

**`regex_error`** if `p` does not contain a valid regular expression pattern interpreted according to `flags`. If **`regex_error`** is thrown, the object remains unchanged.

Definition at line 637 of file `regex.h`.

References `std::basic_regex<_Ch_type, _Rx_traits>::assign()`.

Referenced by `std::basic_regex<_Ch_type, _Rx_traits>::assign()`.

```
5.388.3.5 template<typename _Ch_type, typename _Rx_traits =
 regex_traits<_Ch_type>> basic_regex& std::basic_regex<
 _Ch_type, _Rx_traits>::assign (initializer_list<_Ch_type> __l,
 flag_type __f = regex_constants::ECMAScript) [inline]
```

Assigns a new regular expression to a regex object.

### Parameters

*l* An initializer list representing a regular expression.

## 5.388 std::basic\_regex< \_Ch\_type, \_Rx\_traits > Class Template Reference 2164

---

*flags* Syntax option flags.

### Exceptions

*regex\_error* if *l* does not contain a valid regular expression pattern interpreted according to *flags*. If *regex\_error* is thrown, the object remains unchanged.

Definition at line 652 of file regex.h.

References std::basic\_regex< \_Ch\_type, \_Rx\_traits >::assign().

Referenced by std::basic\_regex< \_Ch\_type, \_Rx\_traits >::assign().

```
5.388.3.6 template<typename _Ch_type, typename _Rx_traits =
 regex_traits<_Ch_type>> basic_regex& std::basic_regex<
 _Ch_type, _Rx_traits >::assign (basic_regex< _Ch_type, _Rx_traits
 > && rhs) [inline]
```

The move-assignment operator.

### Parameters

*rhs* Another regular expression object.

Definition at line 559 of file regex.h.

References std::basic\_regex< \_Ch\_type, \_Rx\_traits >::swap().

```
5.388.3.7 template<typename _Ch_type, typename _Rx_traits =
 regex_traits<_Ch_type>> basic_regex& std::basic_regex<
 _Ch_type, _Rx_traits >::assign (const _Ch_type * p, flag_type
 __flags = regex_constants::ECMAScript) [inline]
```

Assigns a new regular expression to a regex object from a C-style null-terminated string containing a regular expression pattern.

### Parameters

*p* A pointer to a C-style null-terminated string containing a regular expression pattern.

*flags* Syntax option flags.

### Exceptions

*[regex\\_error](#)* if `p` does not contain a valid regular expression pattern interpreted according to `flags`. If *[regex\\_error](#)* is thrown, `*this` remains unchanged.

Definition at line 580 of file `regex.h`.

References `std::basic_regex< _Ch_type, _Rx_traits >::assign()`.

Referenced by `std::basic_regex< _Ch_type, _Rx_traits >::assign()`.

**5.388.3.8** `template<typename _Ch_type, typename _Rx_traits =  
regex_traits<_Ch_type>> flag_type std::basic_regex< _Ch_type,  
_Rx_traits >::flags ( ) const [inline]`

Gets the flags used to construct the regular expression or in the last call to *[assign\(\)](#)*.

Definition at line 670 of file `regex.h`.

Referenced by `std::basic_regex< _Ch_type, _Rx_traits >::operator=()`.

**5.388.3.9** `template<typename _Ch_type, typename _Rx_traits =  
regex_traits<_Ch_type>> locale_type std::basic_regex< _Ch_type,  
_Rx_traits >::getloc ( ) const [inline]`

Gets the locale currently imbued in the regular expression object.

Definition at line 688 of file `regex.h`.

**5.388.3.10** `template<typename _Ch_type, typename _Rx_traits =  
regex_traits<_Ch_type>> locale_type std::basic_regex< _Ch_type,  
_Rx_traits >::imbue ( locale_type __loc ) [inline]`

Imbues the regular expression object with the given locale.

### Parameters

*loc* A locale.

Definition at line 680 of file `regex.h`.



## 5.388 `std::basic_regex<_Ch_type, _Rx_traits>` Class Template Reference 2166

---

**5.388.3.11** `template<typename _Ch_type, typename _Rx_traits =  
regex_traits<_Ch_type>> unsigned int std::basic_regex<  
_Ch_type, _Rx_traits>::mark_count( ) const [inline]`

Gets the number of marked subexpressions within the regular expression.

Definition at line 662 of file `regex.h`.

**5.388.3.12** `template<typename _Ch_type, typename _Rx_traits =  
regex_traits<_Ch_type>> basic_regex& std::basic_regex<  
_Ch_type, _Rx_traits>::operator=( basic_regex<_Ch_type,  
_Rx_traits> && __rhs ) [inline]`

Move-assigns one regular expression to another.

Definition at line 514 of file `regex.h`.

References `std::basic_regex<_Ch_type, _Rx_traits>::assign()`.

**5.388.3.13** `template<typename _Ch_type, typename _Rx_traits =  
regex_traits<_Ch_type>> basic_regex& std::basic_regex<  
_Ch_type, _Rx_traits>::operator=( const basic_regex<_Ch_type,  
_Rx_traits> & __rhs ) [inline]`

Assigns one regular expression to another.

Definition at line 507 of file `regex.h`.

References `std::basic_regex<_Ch_type, _Rx_traits>::assign()`.

**5.388.3.14** `template<typename _Ch_type, typename _Rx_traits =  
regex_traits<_Ch_type>> template<typename _Ch_type_traits ,  
typename _Allocator > basic_regex& std::basic_regex<_Ch_type,  
_Rx_traits>::operator=( const basic_string<_Ch_type,  
_Ch_type_traits, _Allocator> & __s ) [inline]`

Replaces a regular expression with a new one constructed from a string.

### Parameters

A pointer to a string containing a regular expression.

## 5.388 `std::basic_regex< _Ch_type, _Rx_traits >` Class Template Reference 2167

Definition at line 536 of file `regex.h`.

References `std::basic_regex< _Ch_type, _Rx_traits >::assign()`, and `std::basic_regex< _Ch_type, _Rx_traits >::flags()`.

```
5.388.3.15 template<typename _Ch_type, typename _Rx_traits =
 regex_traits<_Ch_type>> basic_regex& std::basic_regex<
 _Ch_type, _Rx_traits >::operator= (const _Ch_type * __p)
 [inline]
```

Replaces a regular expression with a new one constructed from a C-style null-terminated string.

### Parameters

*A* pointer to the start of a null-terminated C-style string containing a regular expression.

Definition at line 525 of file `regex.h`.

References `std::basic_regex< _Ch_type, _Rx_traits >::assign()`, and `std::basic_regex< _Ch_type, _Rx_traits >::flags()`.

```
5.388.3.16 template<typename _Ch_type, typename _Rx_traits =
 regex_traits<_Ch_type>> void std::basic_regex< _Ch_type,
 _Rx_traits >::swap (basic_regex< _Ch_type, _Rx_traits > &
 __rhs) [inline]
```

Swaps the contents of two regular expression objects.

### Parameters

*rhs* Another regular expression object.

Definition at line 698 of file `regex.h`.

Referenced by `std::basic_regex< _Ch_type, _Rx_traits >::assign()`, and `std::swap()`.

The documentation for this class was generated from the following file:

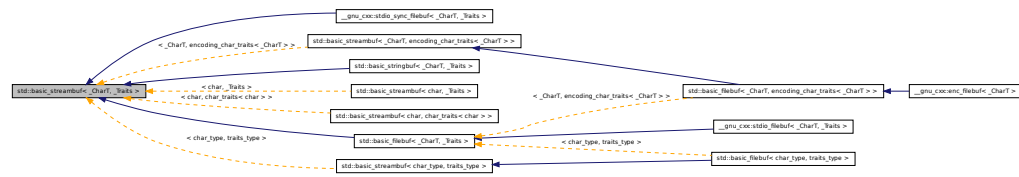
- [regex.h](#)

## 5.389 `std::basic_streambuf< _CharT, _Traits >` Class Template Reference

The actual work of input and output (interface).

This is a base class. Derived stream buffers each control a pair of character sequences: one for input, and one for output.

Inheritance diagram for `std::basic_streambuf< _CharT, _Traits >`:



### Public Types

- `typedef _CharT char_type`
- `typedef _Traits traits_type`
- `typedef traits_type::int_type int_type`
- `typedef traits_type::pos_type pos_type`
- `typedef traits_type::off_type off_type`
- `typedef basic_streambuf< char_type, traits_type > __streambuf_type`

### Public Member Functions

- `streamsize in_avail ()`
- `int_type sbumpc ()`
- `int_type sgetc ()`
- `streamsize sgetn (char_type *__s, streamsize __n)`
- `int_type snextc ()`
- `int_type sputbackc (char_type __c)`
- `int_type sputc (char_type __c)`
- `streamsize sputn (const char_type *__s, streamsize __n)`
- `int_type sungetc ()`

### Protected Member Functions

- `basic_streambuf ()`

- void `gbump` (int \_\_n)
  - virtual void `imbue` (const `locale` &)
  - virtual `int_type` `overflow` (`int_type`=`traits_type::eof()`)
  - virtual `int_type` `pbackfail` (`int_type`=`traits_type::eof()`)
  - void `pbump` (int \_\_n)
  - virtual `pos_type` `seekoff` (`off_type`, `ios_base::seekdir`, `ios_base::openmode`=`ios_base::in|ios_base::out`)
  - virtual `pos_type` `seekpos` (`pos_type`, `ios_base::openmode`=`ios_base::in|ios_base::out`)
  - virtual `basic_streambuf< char_type, _Traits > * setbuf` (`char_type *`, `streamsize`)
  - void `setg` (`char_type *`\_\_gbeg, `char_type *`\_\_gnext, `char_type *`\_\_gend)
  - void `setp` (`char_type *`\_\_pbeg, `char_type *`\_\_pend)
  - virtual `streamsize showmanyc` ()
  - virtual int `sync` ()
  - virtual `int_type` `uflow` ()
  - virtual `int_type` `underflow` ()
  - virtual `streamsize xsgetn` (`char_type *`\_\_s, `streamsize` \_\_n)
  - virtual `streamsize xspn` (const `char_type *`\_\_s, `streamsize` \_\_n)
- 
- `char_type *` `eback` () const
  - `char_type *` `gptr` () const
  - `char_type *` `egptr` () const
- 
- `char_type *` `pbase` () const
  - `char_type *` `pptr` () const
  - `char_type *` `pptr` () const

## Friends

- template<bool \_IsMove, typename \_CharT2 >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_char< \_CharT2 >::\_\_value, \_CharT2 \* >::\_\_type \_\_**copy\_move\_a2** (`istreambuf_iterator< _CharT2 >`, `istreambuf_iterator< _CharT2 >`, `_CharT2 *`)
- `streamsize` \_\_**copy\_streambufs\_eof** (`__streambuf_type *`, `__streambuf_type *`, `bool` &)
- class **basic\_ios**< `char_type`, `traits_type` >
- class **basic\_istream**< `char_type`, `traits_type` >
- class **basic\_ostream**< `char_type`, `traits_type` >
- template<typename \_CharT2 >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_char< \_CharT2 >::\_\_value, `istreambuf_iterator< _CharT2 >` >::\_\_type **find** (`istreambuf_iterator< _CharT2 >`, `istreambuf_iterator< _CharT2 >`, const `_CharT2` &)

## 5.389 std::basic\_streambuf<\_CharT, \_Traits> Class Template Reference 2170

- `template<typename _CharT2, typename _Traits2, typename _Alloc >`  
`basic_istream< _CharT2, _Traits2 > & getline (basic_istream< _CharT2, _Traits2 > &, basic_string< _CharT2, _Traits2, _Alloc > &, _CharT2)`
- `class istreambuf_iterator< char_type, traits_type >`
- `template<typename _CharT2, typename _Traits2, typename _Alloc >`  
`basic_istream< _CharT2, _Traits2 > & operator>> (basic_istream< _CharT2, _Traits2 > &, basic_string< _CharT2, _Traits2, _Alloc > &)`
- `template<typename _CharT2, typename _Traits2 >`  
`basic_istream< _CharT2, _Traits2 > & operator>> (basic_istream< _CharT2, _Traits2 > &, _CharT2 *)`
- `class ostreambuf_iterator< char_type, traits_type >`
- `char_type * _M_in_beg`
- `char_type * _M_in_cur`
- `char_type * _M_in_end`
- `char_type * _M_out_beg`
- `char_type * _M_out_cur`
- `char_type * _M_out_end`
- `locale _M_buf_locale`
- `virtual ~basic_streambuf ()`
- `locale pubimbue (const locale &__loc)`
- `locale getloc () const`
- `__streambuf_type * pubsetbuf (char_type * __s, streamsize __n)`
- `pos_type pubseekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)`
- `pos_type pubseekpos (pos_type __sp, ios_base::openmode __mode=ios_base::in|ios_base::out)`
- `int pubsync ()`

### 5.389.1 Detailed Description

`template<typename _CharT, typename _Traits> class std::basic_streambuf< _CharT, _Traits >`

The actual work of input and output (interface).

This is a base class. Derived stream buffers each control a pair of character sequences: one for input, and one for output. Section [27.5.1] of the standard describes the requirements and behavior of stream buffer classes. That section (three paragraphs) is reproduced here, for simplicity and accuracy.

1. Stream buffers can impose various constraints on the sequences they control. Some constraints are:

- The controlled input sequence can be not readable.
  - The controlled output sequence can be not writable.
  - The controlled sequences can be associated with the contents of other representations for character sequences, such as external files.
  - The controlled sequences can support operations *directly* to or from associated sequences.
  - The controlled sequences can impose limitations on how the program can read characters from a sequence, write characters to a sequence, put characters back into an input sequence, or alter the stream position.
2. Each sequence is characterized by three pointers which, if non-null, all point into the same `charT` array object. The array object represents, at any moment, a (sub)sequence of characters from the sequence. Operations performed on a sequence alter the values stored in these pointers, perform reads and writes directly to or from associated sequences, and alter *the stream position* and conversion state as needed to maintain this subsequence relationship. The three pointers are:
- the *beginning pointer*, or lowest element address in the array (called *xbeg* here);
  - the *next pointer*, or next element address that is a current candidate for reading or writing (called *xnext* here);
  - the *end pointer*, or first element address beyond the end of the array (called *xend* here).
3. The following semantic constraints shall always apply for any set of three pointers for a sequence, using the pointer names given immediately above:
- If *xnext* is not a null pointer, then *xbeg* and *xend* shall also be non-null pointers into the same `charT` array, as described above; otherwise, *xbeg* and *xend* shall also be null.
  - If *xnext* is not a null pointer and  $xnext < xend$  for an output sequence, then a *write position* is available. In this case, *\*xnext* shall be assignable as the next element to write (to put, or to store a character value, into the sequence).
  - If *xnext* is not a null pointer and  $xbeg < xnext$  for an input sequence, then a *putback position* is available. In this case, *xnext[-1]* shall have a defined value and is the next (preceding) element to store a character that is put back into the input sequence.
  - If *xnext* is not a null pointer and  $xnext < xend$  for an input sequence, then a *read position* is available. In this case, *\*xnext* shall have a defined value and is the next element to read (to get, or to obtain a character value, from the sequence).

Definition at line 117 of file `streambuf`.

## 5.389.2 Member Typedef Documentation

**5.389.2.1** `template<typename _CharT, typename _Traits> typedef  
 basic_streambuf<char_type, traits_type> std::basic_streambuf<  
 _CharT, _Traits >::__streambuf_type`

This is a non-standard type.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, and `std::basic_filebuf< char_type, traits_type >`.

Definition at line 135 of file `streambuf`.

**5.389.2.2** `template<typename _CharT, typename _Traits> typedef _CharT  
 std::basic_streambuf< _CharT, _Traits >::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, and `std::basic_filebuf< char_type, traits_type >`.

Definition at line 126 of file `streambuf`.

**5.389.2.3** `template<typename _CharT, typename _Traits> typedef  
 traits_type::int_type std::basic_streambuf< _CharT, _Traits  
 >::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, and `std::basic_filebuf< char_type, traits_type >`.

Definition at line 128 of file `streambuf`.

**5.389.2.4** `template<typename _CharT, typename _Traits> typedef  
traits_type::off_type std::basic_streambuf<_CharT, _Traits  
>::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in [std::basic\\_filebuf<\\_CharT, \\_Traits>](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>](#), [\\_\\_gnu\\_cxx::stdio\\_filebuf<\\_CharT, \\_Traits>](#), [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf<\\_CharT, \\_Traits>](#), [std::basic\\_filebuf<\\_CharT, encoding\\_char\\_traits<\\_CharT>>](#), and [std::basic\\_filebuf<char\\_type, traits\\_type>](#).

Definition at line 130 of file `streambuf`.

**5.389.2.5** `template<typename _CharT, typename _Traits> typedef  
traits_type::pos_type std::basic_streambuf<_CharT, _Traits  
>::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in [std::basic\\_filebuf<\\_CharT, \\_Traits>](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>](#), [\\_\\_gnu\\_cxx::enc\\_filebuf<\\_CharT>](#), [\\_\\_gnu\\_cxx::stdio\\_filebuf<\\_CharT, \\_Traits>](#), [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf<\\_CharT, \\_Traits>](#), [std::basic\\_filebuf<\\_CharT, encoding\\_char\\_traits<\\_CharT>>](#), and [std::basic\\_filebuf<char\\_type, traits\\_type>](#).

Definition at line 129 of file `streambuf`.

**5.389.2.6** `template<typename _CharT, typename _Traits> typedef _Traits  
std::basic_streambuf<_CharT, _Traits>::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in [std::basic\\_filebuf<\\_CharT, \\_Traits>](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>](#), [\\_\\_gnu\\_cxx::enc\\_filebuf<\\_CharT>](#), [\\_\\_gnu\\_cxx::stdio\\_filebuf<\\_CharT, \\_Traits>](#), [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf<\\_CharT, \\_Traits>](#), [std::basic\\_filebuf<\\_CharT, encoding\\_char\\_traits<\\_CharT>>](#), and [std::basic\\_filebuf<char\\_type, traits\\_type>](#).

Definition at line 127 of file `streambuf`.



### **5.389.3 Constructor & Destructor Documentation**

**5.389.3.1** `template<typename _CharT, typename _Traits> virtual  
std::basic_streambuf<_CharT, _Traits>::~~basic_streambuf ( )  
[inline, virtual]`

Destructor deallocates no buffer space.

Definition at line 195 of file streambuf.

**5.389.3.2** `template<typename _CharT, typename _Traits>  
std::basic_streambuf<_CharT, _Traits>::basic_streambuf ( )  
[inline, protected]`

Base constructor.

Only called from derived constructors, and sets up all the buffer data to zero, including the pointers described in the [basic\\_streambuf](#) class description. Note that, as a result,

- the class starts with no read nor write positions available,
- this is not an error

Definition at line 443 of file streambuf.

### **5.389.4 Member Function Documentation**

**5.389.4.1** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::eback ( ) const  
[inline, protected]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- [eback\(\)](#) returns the beginning pointer for the input sequence
- [gptr\(\)](#) returns the next pointer for the input sequence
- [egptr\(\)](#) returns the end pointer for the input sequence

Definition at line 462 of file streambuf.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::\_M\_destroy\_pback(), std::basic\_filebuf< \_CharT, \_Traits >::imbue(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::overflow(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::pbackfail(), std::basic\_filebuf< \_CharT, \_Traits >::pbackfail(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos(), std::basic\_streambuf< char\_type, traits\_type >::sputbackc(), std::basic\_streambuf< char\_type, traits\_type >::sungetc(), std::basic\_filebuf< \_CharT, \_Traits >::underflow(), and std::basic\_filebuf< \_CharT, \_Traits >::xsgetn().

**5.389.4.2** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::egptr ( ) const  
[inline, protected]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- [eback\(\)](#) returns the beginning pointer for the input sequence
- [gptr\(\)](#) returns the next pointer for the input sequence
- [egptr\(\)](#) returns the end pointer for the input sequence

Definition at line 468 of file streambuf.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::\_M\_create\_pback(), std::basic\_streambuf< char\_type, traits\_type >::in\_avail(), std::basic\_streambuf< char\_type, traits\_type >::sbumpc(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos(), std::basic\_streambuf< char\_type, traits\_type >::sgetc(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::showmanyc(), std::basic\_filebuf< \_CharT, \_Traits >::showmanyc(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::str(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::underflow(), std::basic\_filebuf< \_CharT, \_Traits >::underflow(), std::basic\_streambuf< \_CharT, \_Traits >::xsgetn(), and std::basic\_filebuf< \_CharT, \_Traits >::xsgetn().

**5.389.4.3** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::eptr ( ) const  
[inline, protected]`

## 5.389 `std::basic_streambuf<_CharT, _Traits>` Class Template Reference 2176

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 515 of file `streambuf`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_streambuf<char_type, traits_type>::sputc()`, `std::basic_streambuf<_CharT, _Traits>::xsputn()`, and `std::basic_filebuf<_CharT, _Traits>::xsputn()`.

**5.389.4.4** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf<_CharT, _Traits>::gbump ( int __n )  
[inline, protected]`

Moving the read position.

### Parameters

*n* The delta by which to move.

This just advances the read position without returning any data.

Definition at line 478 of file `streambuf`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::pbackfail()`, `std::basic_filebuf<_CharT, _Traits>::pbackfail()`, `std::basic_streambuf<char_type, traits_type>::sbumpc()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, `std::basic_streambuf<char_type, traits_type>::sputbackc()`, `std::basic_streambuf<char_type, traits_type>::sungetc()`, `std::basic_streambuf<char_type, traits_type>::uflow()`, `std::basic_streambuf<_CharT, _Traits>::xsgetn()`, and `std::basic_filebuf<_CharT, _Traits>::xsgetn()`.

**5.389.4.5** `template<typename _CharT, typename _Traits> locale  
std::basic_streambuf<_CharT, _Traits>::getloc ( ) const  
[inline]`

Locale access.

**Returns**

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 224 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::pubimbue()`.

**5.389.4.6** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::gptr ( ) const  
[inline, protected]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 465 of file `streambuf`.

Referenced by `std::basic_filebuf<char_type, traits_type>::_M_create_pback()`, `std::basic_filebuf<char_type, traits_type>::_M_destroy_pback()`, `std::basic_filebuf<_CharT, _Traits>::imbue()`, `std::basic_streambuf<char_type, traits_type>::in_avail()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::pbackfail()`, `std::basic_filebuf<_CharT, _Traits>::pbackfail()`, `std::basic_streambuf<char_type, traits_type>::sbumpc()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, `std::basic_streambuf<char_type, traits_type>::sgetc()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::showmanyc()`, `std::basic_filebuf<_CharT, _Traits>::showmanyc()`, `std::basic_streambuf<char_type, traits_type>::sputbackc()`, `std::basic_streambuf<char_type, traits_type>::sungetc()`, `std::basic_streambuf<char_type, traits_type>::uflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::underflow()`, `std::basic_filebuf<_CharT, _Traits>::underflow()`, `std::basic_streambuf<_CharT, _Traits>::xsgetn()`, and `std::basic_filebuf<_CharT, _Traits>::xsgetn()`.

**5.389.4.7** `template<typename _CharT, typename _Traits> virtual void  
std::basic_streambuf<_CharT, _Traits>::imbue ( const locale & )  
[inline, protected, virtual]`

Changes translations.

#### Parameters

*loc* A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

#### Note

Base class version does nothing.

Reimplemented in [std::basic\\_filebuf<\\_CharT, \\_Traits>](#), [std::basic\\_filebuf<\\_CharT, encoding\\_char\\_traits<\\_CharT>>](#), and [std::basic\\_filebuf<char\\_type, traits\\_type>](#).

Definition at line 556 of file streambuf.

Referenced by `std::basic_streambuf<char_type, traits_type>::pubimbue()`.

**5.389.4.8** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf<_CharT, _Traits>::in_avail ( ) [inline]`

Looking ahead into the stream.

#### Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived [showmanyc\(\)](#).

Definition at line 264 of file streambuf.

**5.389.4.9** `template<typename _CharT, typename _Traits> virtual int_type  
std::basic_streambuf<_CharT, _Traits>::overflow ( int_type =  
traits_type::eof() ) [inline, protected, virtual]`

Consumes data from the buffer; writes to the controlled sequence.

#### Parameters

*c* An additional character to consume.

#### Returns

`eof()` to indicate failure, something else (usually *c*, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if *c* is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

#### Note

Base class version does nothing, returns `eof()`.

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>`, `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, and `std::basic_filebuf<char_type, traits_type>`.

Definition at line 748 of file streambuf.

Referenced by `std::basic_streambuf<char_type, traits_type>::sputc()`, and `std::basic_streambuf<_CharT, _Traits>::xsputn()`.

```
5.389.4.10 template<typename _CharT, typename _Traits> virtual int_type
std::basic_streambuf<_CharT, _Traits>::pbackfail (int_type =
traits_type::eof()) [inline, protected, virtual]
```

Tries to back up the input sequence.

#### Parameters

*c* The character to be inserted back into the sequence.

#### Returns

`eof()` on failure, *some other value* on success

## 5.389 std::basic\_streambuf<\_CharT, \_Traits> Class Template Reference 2180

### Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

### Note

Base class version does nothing, returns eof().

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>`, `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, and `std::basic_filebuf<char_type, traits_type>`.

Definition at line 704 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::sputbackc()`, and `std::basic_streambuf<char_type, traits_type>::sungetc()`.

**5.389.4.11** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::pbase ( ) const  
[inline, protected]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 509 of file `streambuf`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_filebuf<_CharT, _Traits>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::str()`, `std::basic_filebuf<_CharT, _Traits>::sync()`, and `std::basic_filebuf<_CharT, _Traits>::xsputn()`.

**5.389.4.12** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf<_CharT, _Traits>::pbump ( int __n )  
[inline, protected]`

Moving the write position.

#### Parameters

- n* The delta by which to move.

This just advances the write position without returning any data.

Definition at line 525 of file streambuf.

Referenced by std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::overflow(), std::basic\_filebuf< \_CharT, \_Traits >::overflow(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos(), std::basic\_streambuf< char\_type, traits\_type >::putc(), and std::basic\_streambuf< \_CharT, \_Traits >::xsputn().

**5.389.4.13** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::pptr ( ) const  
[inline, protected]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 512 of file streambuf.

Referenced by std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::overflow(), std::basic\_filebuf< \_CharT, \_Traits >::overflow(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_filebuf< \_CharT, \_Traits >::seekoff(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos(), std::basic\_streambuf< char\_type, traits\_type >::putc(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::str(), std::basic\_filebuf< \_CharT, \_Traits >::sync(), std::basic\_streambuf< \_CharT, \_Traits >::xsputn(), and std::basic\_filebuf< \_CharT, \_Traits >::xsputn().

**5.389.4.14** `template<typename _CharT, typename _Traits> locale  
std::basic_streambuf< _CharT, _Traits >::pubimbue ( const locale  
& __loc ) [inline]`



## 5.389 std::basic\_streambuf<\_CharT, \_Traits> Class Template Reference 2182

---

Entry point for [imbue\(\)](#).

### Parameters

*loc* The new locale.

### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 207 of file `streambuf`.

**5.389.4.15** `template<typename _CharT, typename _Traits> pos_type  
std::basic_streambuf<_CharT, _Traits>::pubseekoff ( off_type  
__off, ios_base::seekdir __way, ios_base::openmode __mode =  
ios_base::in | ios_base::out ) [inline]`

Entry point for [imbue\(\)](#).

### Parameters

*loc* The new locale.

### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 241 of file `streambuf`.

**5.389.4.16** `template<typename _CharT, typename _Traits> pos_type  
std::basic_streambuf<_CharT, _Traits>::pubseekpos ( pos_type  
__sp, ios_base::openmode __mode = ios_base::in | ios_base::out )  
[inline]`

Entry point for [imbue\(\)](#).

### Parameters

*loc* The new locale.

## 5.389 std::basic\_streambuf<\_CharT, \_Traits> Class Template Reference 2183

---

### Returns

The previous locale.

Calls the derived imbue(loc).

Definition at line 246 of file streambuf.

**5.389.4.17** `template<typename _CharT, typename _Traits>  
__streambuf_type* std::basic_streambuf<_CharT, _Traits  
>::pubsetbuf ( char_type * __s, streamsize __n ) [inline]`

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 237 of file streambuf.

**5.389.4.18** `template<typename _CharT, typename _Traits> int  
std::basic_streambuf<_CharT, _Traits>::pubsync ( )  
[inline]`

Entry point for [imbue\(\)](#).

### Parameters

*loc* The new locale.

### Returns

The previous locale.

Calls the derived imbue(loc).

Definition at line 251 of file streambuf.

Referenced by `std::basic_istream<_CharT, _Traits>::sync()`.

**5.389.4.19** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf<_CharT, _Traits>::sbumpc ( ) [inline]`

Getting the next character.

**Returns**

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 296 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::istreambuf_iterator<_CharT, _Traits>::operator++()`, and `std::basic_streambuf<char_type, traits_type>::snextc()`.

**5.389.4.20** `template<typename _CharT, typename _Traits> virtual pos_type std::basic_streambuf<_CharT, _Traits>::seekoff ( off_type, ios_base::seekdir, ios_base::openmode = ios_base::in | ios_base::out ) [inline, protected, virtual]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, and `std::basic_filebuf<char_type, traits_type>`.

Definition at line 582 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::pubseekoff()`.

**5.389.4.21** `template<typename _CharT, typename _Traits> virtual pos_type std::basic_streambuf<_CharT, _Traits>::seekpos ( pos_type, ios_base::openmode = ios_base::in | ios_base::out ) [inline, protected, virtual]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

## 5.389 `std::basic_streambuf< _CharT, _Traits >` Class Template Reference 2185

### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, and `std::basic_filebuf< char_type, traits_type >`.

Definition at line 594 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::pubseekpos()`.

```
5.389.4.22 template<typename _CharT, typename _Traits> virtual
 basic_streambuf<char_type, _Traits>* std::basic_streambuf<
 _CharT, _Traits >::setbuf (char_type *, streamsize) [inline,
 protected, virtual]
```

Manipulates the buffer.

Each derived class provides its own appropriate behavior. See the next-to-last paragraph of <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more on this function.

### Note

Base class version does nothing, returns `this`.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, and `std::basic_filebuf< char_type, traits_type >`.

Definition at line 571 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::pubsetbuf()`.

```
5.389.4.23 template<typename _CharT, typename _Traits> void
 std::basic_streambuf< _CharT, _Traits >::setg (char_type *
 __gbeg, char_type * __gnext, char_type * __gend) [inline,
 protected]
```

Setting the three read area pointers.

### Parameters

*gbeg* A pointer.

*gnext* A pointer.

*gend* A pointer.

**Postcondition**

*gbeg* == `eback()`, *gnext* == `gptr()`, and *gend* == `egptr()`

Definition at line 489 of file streambuf.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_create_pback()`, `std::basic_filebuf< char_type, traits_type >::_M_destroy_pback()`, and `std::basic_filebuf< char_type, traits_type >::_M_set_buffer()`.

**5.389.4.24** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf< _CharT, _Traits >::setp ( char_type *  
__pbeg, char_type * __pend ) [inline, protected]`

Setting the three write area pointers.

**Parameters**

*pbeg* A pointer.

*pend* A pointer.

**Postcondition**

*pbeg* == `pbase()`, *pbeg* == `pptr()`, and *pend* == `pptr()`

Definition at line 535 of file streambuf.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_set_buffer()`.

**5.389.4.25** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf< _CharT, _Traits >::sgetc ( ) [inline]`

Getting the next character.

**Returns**

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

## 5.389 std::basic\_streambuf<\_CharT, \_Traits> Class Template Reference 2187

---

Definition at line 318 of file streambuf.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, and `std::basic_streambuf<char_type, traits_type>::snextc()`.

**5.389.4.26** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf<_CharT, _Traits>::sgetn ( char_type * __s,  
streamsize __n ) [inline]`

Entry point for `xsgetn`.

### Parameters

*s* A buffer area.

*n* A count.

Returns `xsgetn(s,n)`. The effect is to fill `s[0]` through `s[n-1]` with characters from the input sequence, if possible.

Definition at line 337 of file streambuf.

**5.389.4.27** `template<typename _CharT, typename _Traits> virtual streamsize  
std::basic_streambuf<_CharT, _Traits>::showmanyc ( )  
[inline, protected, virtual]`

Investigating the data available.

### Returns

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.* [27.5.2.4.3]/1

### Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

## 5.389 std::basic\_streambuf<\_CharT, \_Traits> Class Template Reference 2188

---

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented in [std::basic\\_filebuf<\\_CharT, \\_Traits>](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>](#), [std::basic\\_filebuf<\\_CharT, encoding\\_char\\_traits<\\_CharT>>](#), and [std::basic\\_filebuf<char\\_type, traits\\_type>](#).

Definition at line 629 of file streambuf.

Referenced by [std::basic\\_streambuf<char\\_type, traits\\_type>::in\\_avail\(\)](#).

**5.389.4.28** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf<_CharT, _Traits>::snextc ( ) [inline]`

Getting the next character.

### Returns

The next character, or eof.

Calls [sbumpc\(\)](#), and if that function returns `traits::eof()`, so does this function. Otherwise, [sgetc\(\)](#).

Definition at line 278 of file streambuf.

Referenced by [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::getline\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), and [std::basic\\_istream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#).

**5.389.4.29** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf<_CharT, _Traits>::sputbackc ( char_type  
__c ) [inline]`

Pushing characters back into the input stream.

### Parameters

*c* The character to push back.

### Returns

The previous character, if possible.

Similar to [sungetc\(\)](#), but *c* is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be *c*.

## 5.389 `std::basic_streambuf<_CharT, _Traits>` Class Template Reference 2189

---

Definition at line 352 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::putback()`.

**5.389.4.30** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf<_CharT, _Traits>::sputc ( char_type __c )  
[inline]`

Entry point for all single-character output functions.

### Parameters

*c* A character to output.

### Returns

*c*, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores *c* in that position, increments the position, and returns `traits::to_int_type(c)`. If a write position is not available, returns `overflow(c)`.

Definition at line 404 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, and `std::ostreambuf_iterator<_CharT, _Traits>::operator=()`.

**5.389.4.31** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf<_CharT, _Traits>::sputn ( const char_type  
* __s, streamsize __n ) [inline]`

Entry point for all single-character output functions.

### Parameters

*s* A buffer read area.

*n* A count.

One of two public output functions.

Returns `xspn(s,n)`. The effect is to write *s*[0] through *s*[*n*-1] to the output sequence, if possible.

Definition at line 430 of file `streambuf`.



## 5.389 `std::basic_streambuf<_CharT, _Traits>` Class Template Reference 2190

**5.389.4.32** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf<_CharT, _Traits>::sungetc ( ) [inline]`

Moving backwards in the input stream.

### Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbackfail()`. The effect is to *unget* the last character *gotten*.

Definition at line 377 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::unget()`.

**5.389.4.33** `template<typename _CharT, typename _Traits> virtual int  
std::basic_streambuf<_CharT, _Traits>::sync ( void )  
[inline, protected, virtual]`

Synchronizes the buffer arrays with the controlled sequences.

### Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

### Note

Base class version does nothing, returns zero.

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, and `std::basic_filebuf<char_type, traits_type>`.

Definition at line 607 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::pubsync()`.

**5.389.4.34** `template<typename _CharT, typename _Traits> virtual int_type  
std::basic_streambuf< _CharT, _Traits >::uflow ( ) [inline,  
protected, virtual]`

Fetches more data from the controlled sequence.

#### Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`.

Definition at line 680 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::sbumpc()`, and `std::basic_streambuf< _CharT, _Traits >::xsgetn()`.

**5.389.4.35** `template<typename _CharT, typename _Traits> virtual int_type  
std::basic_streambuf< _CharT, _Traits >::underflow ( )  
[inline, protected, virtual]`

Fetches more data from the controlled sequence.

#### Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25.html>

#### Note

Base class version does nothing, returns `eof()`.

## 5.389 `std::basic_streambuf<_CharT, _Traits>` Class Template Reference 2192

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>`, `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, and `std::basic_filebuf<char_type, traits_type>`.

Definition at line 667 of file `streambuf`.

Referenced by `std::basic_streambuf<char_type, traits_type>::sgetc()`, and `std::basic_streambuf<char_type, traits_type>::uflow()`.

**5.389.4.36** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf<_CharT, _Traits>::xsgetn ( char_type *  
__s, streamsize __n ) [protected, virtual]`

Multiple character extraction.

### Parameters

*s* A buffer area.

*n* Maximum number of characters to assign.

### Returns

The number of characters assigned.

Fills `s[0]` through `s[n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either *n* characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, and `std::basic_filebuf<char_type, traits_type>`.

Definition at line 47 of file `streambuf.tcc`.

References `std::basic_streambuf<_CharT, _Traits>::egptr()`, `std::basic_streambuf<_CharT, _Traits>::gbump()`, `std::basic_streambuf<_CharT, _Traits>::gptr()`, `std::min()`, and `std::basic_streambuf<_CharT, _Traits>::uflow()`.

Referenced by `std::basic_streambuf<char_type, traits_type>::xsgetn()`.

**5.389.4.37** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf<_CharT, _Traits>::xsputn ( const char_type  
* __s, streamsize __n ) [protected, virtual]`

Multiple character insertion.

#### Parameters

- s* A buffer area.
- n* Maximum number of characters to write.

#### Returns

The number of characters written.

Writes *s*[0] through *s*[*n*-1] to the output sequence, as if by `sputc()`. Stops when either *n* characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, and `std::basic_filebuf<char_type, traits_type>`.

Definition at line 81 of file `streambuf.tcc`.

References `std::basic_streambuf<_CharT, _Traits>::eptr()`, `std::min()`, `std::basic_streambuf<_CharT, _Traits>::overflow()`, `std::basic_streambuf<_CharT, _Traits>::pbump()`, and `std::basic_streambuf<_CharT, _Traits>::pptr()`.

Referenced by `std::basic_streambuf<char_type, traits_type>::sputn()`.

### 5.389.5 Member Data Documentation

**5.389.5.1** `template<typename _CharT, typename _Traits> locale  
std::basic_streambuf<_CharT, _Traits>::_M_buf_locale  
[protected]`

Current locale setting.

Definition at line 190 of file `streambuf`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::basic_filebuf()`, `std::basic_streambuf<char_type, traits_type>::getloc()`, and `std::basic_streambuf<char_type, traits_type>::pubimbue()`.

**5.389.5.2** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_in_beg  
[protected]`

This is based on `_IO_FILE`, just reordered to be more consistent, and is intended to be the most minimal abstraction for an internal buffer.

- `get == input == read`
- `put == output == write`

Definition at line 182 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::eback()`, and `std::basic_streambuf< char_type, traits_type >::setg()`.

**5.389.5.3** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_in_cur  
[protected]`

Entry point for `imbue()`.

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 183 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::gbump()`, `std::basic_streambuf< char_type, traits_type >::gptr()`, and `std::basic_streambuf< char_type, traits_type >::setg()`.

**5.389.5.4** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_in_end  
[protected]`

Entry point for `imbue()`.

**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 184 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::egptr()`, and `std::basic_streambuf< char_type, traits_type >::setg()`.

**5.389.5.5** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_out_beg  
[protected]`

Entry point for `imbue()`.

**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 185 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::pbase()`, and `std::basic_streambuf< char_type, traits_type >::setp()`.

**5.389.5.6** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_out_cur  
[protected]`

Entry point for `imbue()`.

**Parameters**

*loc* The new locale.

### Returns

The previous locale.

Calls the derived imbue(loc).

Definition at line 186 of file streambuf.

Referenced by std::basic\_streambuf< char\_type, traits\_type >::pbump(), std::basic\_streambuf< char\_type, traits\_type >::pptr(), and std::basic\_streambuf< char\_type, traits\_type >::setp().

**5.389.5.7 template<typename \_CharT, typename \_Traits> char\_type\*  
std::basic\_streambuf< \_CharT, \_Traits >::M\_out\_end  
[protected]**

Entry point for [imbue\(\)](#).

### Parameters

*loc* The new locale.

### Returns

The previous locale.

Calls the derived imbue(loc).

Definition at line 187 of file streambuf.

Referenced by std::basic\_streambuf< char\_type, traits\_type >::epptr(), and std::basic\_streambuf< char\_type, traits\_type >::setp().

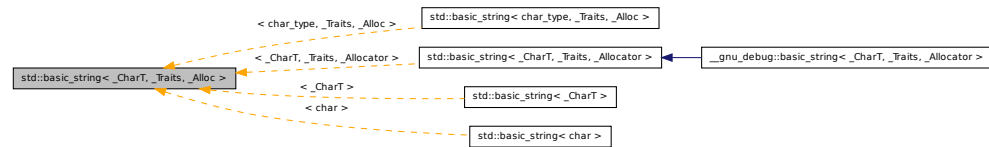
The documentation for this class was generated from the following files:

- [streambuf](#)
- [streambuf.tcc](#)

## 5.390 std::basic\_string<\_CharT, \_Traits, \_Alloc> Class Template Reference

Managing sequences of characters and character-like objects.

Inheritance diagram for `std::basic_string<_CharT, _Traits, _Alloc>`:



## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `__gnu_cxx::__normal_iterator< const_pointer, basic_string >` **const\_iterator**
- typedef `_CharT_alloc_type::const_pointer` **const\_pointer**
- typedef `_CharT_alloc_type::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_CharT_alloc_type::difference_type` **difference\_type**
- typedef `__gnu_cxx::__normal_iterator< pointer, basic_string >` **iterator**
- typedef `_CharT_alloc_type::pointer` **pointer**
- typedef `_CharT_alloc_type::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `_CharT_alloc_type::size_type` **size\_type**
- typedef `_Traits` **traits\_type**
- typedef `_Traits::char_type` **value\_type**

## Public Member Functions

- `basic_string()`
- `basic_string(const _Alloc &__a)`
- `basic_string(const basic_string &__str, size_type __pos, size_type __n=npos)`
- `basic_string(size_type __n, _CharT __c, const _Alloc &__a=_Alloc())`
- `basic_string(basic_string &&__str)`
- `basic_string(const basic_string &__str, size_type __pos, size_type __n, const _Alloc &__a)`
- `basic_string(initializer_list< _CharT > __l, const _Alloc &__a=_Alloc())`
- `template<class _InputIterator>`  
`basic_string(_InputIterator __beg, _InputIterator __end, const _Alloc &__a=_Alloc())`
- `basic_string(const basic_string &__str)`
- `basic_string(const _CharT * __s, size_type __n, const _Alloc &__a=_Alloc())`



- `basic_string` (const `_CharT *__s`, const `_Alloc &__a=_Alloc()`)
- `~basic_string` ()
- `template<typename _InIterator >`  
`_CharT * S_construct (_InIterator __beg, _InIterator __end, const _Alloc &__a, forward_iterator_tag)`
- `basic_string & append` (const `basic_string &__str`)
- `basic_string & append` (const `basic_string &__str`, `size_type __pos`, `size_type __n`)
- `basic_string & append` (const `_CharT *__s`, `size_type __n`)
- `basic_string & append` (const `_CharT *__s`)
- `basic_string & append` (`size_type __n`, `_CharT __c`)
- `basic_string & append` (`initializer_list<_CharT> __l`)
- `template<class _InputIterator >`  
`basic_string & append` (`_InputIterator __first`, `_InputIterator __last`)
- `basic_string & assign` (const `_CharT *__s`)
- `basic_string & assign` (`size_type __n`, `_CharT __c`)
- `template<class _InputIterator >`  
`basic_string & assign` (`_InputIterator __first`, `_InputIterator __last`)
- `basic_string & assign` (`initializer_list<_CharT> __l`)
- `basic_string & assign` (const `basic_string &__str`)
- `basic_string & assign` (`basic_string &&__str`)
- `basic_string & assign` (const `basic_string &__str`, `size_type __pos`, `size_type __n`)
- `basic_string & assign` (const `_CharT *__s`, `size_type __n`)
- `const_reference at` (`size_type __n`) const
- `reference at` (`size_type __n`)
- `reference back` ()
- `const_reference back` () const
- `iterator begin` ()
- `const_iterator begin` () const
- `const _CharT * c_str` () const
- `size_type capacity` () const
- `const_iterator cbegin` () const
- `const_iterator cend` () const
- `void clear` ()
- `int compare` (`size_type __pos`, `size_type __n1`, const `_CharT *__s`) const
- `int compare` (const `_CharT *__s`) const
- `int compare` (`size_type __pos`, `size_type __n1`, const `_CharT *__s`, `size_type __n2`) const
- `int compare` (`size_type __pos`, `size_type __n`, const `basic_string &__str`) const
- `int compare` (`size_type __pos1`, `size_type __n1`, const `basic_string &__str`, `size_type __pos2`, `size_type __n2`) const
- `int compare` (const `basic_string &__str`) const

- `size_type copy` (`_CharT *__s`, `size_type __n`, `size_type __pos=0`) `const`
- `const_reverse_iterator crbegin` () `const`
- `const_reverse_iterator crend` () `const`
- `const _CharT * data` () `const`
- `bool empty` () `const`
- `iterator end` ()
- `const_iterator end` () `const`
- `basic_string & erase` (`size_type __pos=0`, `size_type __n=npos`)
- `iterator erase` (`iterator __position`)
- `iterator erase` (`iterator __first`, `iterator __last`)
- `size_type find` (`_CharT __c`, `size_type __pos=0`) `const`
- `size_type find` (`const _CharT *__s`, `size_type __pos`, `size_type __n`) `const`
- `size_type find` (`const basic_string &__str`, `size_type __pos=0`) `const`
- `size_type find` (`const _CharT *__s`, `size_type __pos=0`) `const`
- `size_type find_first_not_of` (`const _CharT *__s`, `size_type __pos`, `size_type __n`) `const`
- `size_type find_first_not_of` (`_CharT __c`, `size_type __pos=0`) `const`
- `size_type find_first_not_of` (`const basic_string &__str`, `size_type __pos=0`) `const`
- `size_type find_first_not_of` (`const _CharT *__s`, `size_type __pos=0`) `const`
- `size_type find_first_of` (`_CharT __c`, `size_type __pos=0`) `const`
- `size_type find_first_of` (`const _CharT *__s`, `size_type __pos`, `size_type __n`) `const`
- `size_type find_first_of` (`const _CharT *__s`, `size_type __pos=0`) `const`
- `size_type find_first_of` (`const basic_string &__str`, `size_type __pos=0`) `const`
- `size_type find_last_not_of` (`const _CharT *__s`, `size_type __pos`, `size_type __n`) `const`
- `size_type find_last_not_of` (`_CharT __c`, `size_type __pos=npo`) `const`
- `size_type find_last_not_of` (`const basic_string &__str`, `size_type __pos=npo`) `const`
- `size_type find_last_not_of` (`const _CharT *__s`, `size_type __pos=npo`) `const`
- `size_type find_last_of` (`const basic_string &__str`, `size_type __pos=npo`) `const`
- `size_type find_last_of` (`_CharT __c`, `size_type __pos=npo`) `const`
- `size_type find_last_of` (`const _CharT *__s`, `size_type __pos=npo`) `const`
- `size_type find_last_of` (`const _CharT *__s`, `size_type __pos`, `size_type __n`) `const`
- `reference front` ()
- `const_reference front` () `const`
- `allocator_type get_allocator` () `const`
- `void insert` (`iterator __p`, `size_type __n`, `_CharT __c`)
- `template<class _InputIterator>`  
  `void insert` (`iterator __p`, `_InputIterator __beg`, `_InputIterator __end`)
- `void insert` (`iterator __p`, `initializer_list<_CharT> __l`)

- `basic_string` & `insert` (size\_type \_\_pos1, const `basic_string` &\_\_str, size\_type \_\_pos2, size\_type \_\_n)
- `basic_string` & `insert` (size\_type \_\_pos, const \_CharT \*\_\_s, size\_type \_\_n)
- `basic_string` & `insert` (size\_type \_\_pos, const \_CharT \*\_\_s)
- iterator `insert` (iterator \_\_p, \_CharT \_\_c)
- `basic_string` & `insert` (size\_type \_\_pos, size\_type \_\_n, \_CharT \_\_c)
- `basic_string` & `insert` (size\_type \_\_pos1, const `basic_string` &\_\_str)
- size\_type `length` () const
- size\_type `max_size` () const
- `basic_string` & `operator+=` (initializer\_list< \_CharT > \_\_l)
- `basic_string` & `operator+=` (const `basic_string` &\_\_str)
- `basic_string` & `operator+=` (\_CharT \_\_c)
- `basic_string` & `operator+=` (const \_CharT \*\_\_s)
- `basic_string` & `operator=` (const \_CharT \*\_\_s)
- `basic_string` & `operator=` (\_CharT \_\_c)
- `basic_string` & `operator=` (initializer\_list< \_CharT > \_\_l)
- `basic_string` & `operator=` (`basic_string` &&\_\_str)
- `basic_string` & `operator=` (const `basic_string` &\_\_str)
- const\_reference `operator[]` (size\_type \_\_pos) const
- reference `operator[]` (size\_type \_\_pos)
- void `push_back` (\_CharT \_\_c)
- `reverse_iterator` `rbegin` ()
- const `reverse_iterator` `rbegin` () const
- `reverse_iterator` `rend` ()
- const `reverse_iterator` `rend` () const
- `basic_string` & `replace` (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \*\_\_s, size\_type \_\_n2)
- `basic_string` & `replace` (iterator \_\_i1, iterator \_\_i2, size\_type \_\_n, \_CharT \_\_c)
- `basic_string` & `replace` (size\_type \_\_pos, size\_type \_\_n1, size\_type \_\_n2, \_CharT \_\_c)
- `basic_string` & `replace` (iterator \_\_i1, iterator \_\_i2, const `basic_string` &\_\_str)
- template<class \_InputIterator >  
`basic_string` & `replace` (iterator \_\_i1, iterator \_\_i2, \_InputIterator \_\_k1, \_InputIterator \_\_k2)
- `basic_string` & `replace` (size\_type \_\_pos, size\_type \_\_n, const `basic_string` &\_\_str)
- `basic_string` & `replace` (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \*\_\_s)
- `basic_string` & `replace` (iterator \_\_i1, iterator \_\_i2, const\_iterator \_\_k1, const\_iterator \_\_k2)
- `basic_string` & `replace` (iterator \_\_i1, iterator \_\_i2, iterator \_\_k1, iterator \_\_k2)
- `basic_string` & `replace` (iterator \_\_i1, iterator \_\_i2, \_CharT \*\_\_k1, \_CharT \*\_\_k2)

- `basic_string` & `replace` (size\_type \_\_pos1, size\_type \_\_n1, const `basic_string` &\_\_str, size\_type \_\_pos2, size\_type \_\_n2)
- `basic_string` & `replace` (iterator \_\_i1, iterator \_\_i2, const \_CharT \*\_\_s)
- `basic_string` & `replace` (iterator \_\_i1, iterator \_\_i2, `initializer_list`< \_CharT > \_\_l)
- `basic_string` & `replace` (iterator \_\_i1, iterator \_\_i2, const \_CharT \*\_\_k1, const \_CharT \*\_\_k2)
- `basic_string` & `replace` (iterator \_\_i1, iterator \_\_i2, const \_CharT \*\_\_s, size\_type \_\_n)
- void `reserve` (size\_type \_\_res\_arg=0)
- void `resize` (size\_type \_\_n)
- void `resize` (size\_type \_\_n, \_CharT \_\_c)
- size\_type `rfind` (\_CharT \_\_c, size\_type \_\_pos=`npos`) const
- size\_type `rfind` (const \_CharT \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const
- size\_type `rfind` (const `basic_string` &\_\_str, size\_type \_\_pos=`npos`) const
- size\_type `rfind` (const \_CharT \*\_\_s, size\_type \_\_pos=`npos`) const
- void `shrink_to_fit` ()
- size\_type `size` () const
- `basic_string` `substr` (size\_type \_\_pos=0, size\_type \_\_n=`npos`) const
- void `swap` (`basic_string` &\_\_s)

### Static Public Attributes

- static const size\_type `npos`

#### 5.390.1 Detailed Description

`template<typename _CharT, typename _Traits, typename _Alloc> class std::basic_string<_CharT, _Traits, _Alloc>`

Managing sequences of characters and character-like objects. Meets the requirements of a `container`, a `reversible container`, and a `sequence`. Of the `optional sequence requirements`, only `push_back`, `at`, and `array access` are supported.

### Todo

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

Documentation? What's that? Nathan Myers <[ncm@cantrip.org](mailto:ncm@cantrip.org)>.

A string looks like this:

|                           |                            |
|---------------------------|----------------------------|
|                           | [_Rep]                     |
|                           | _M_length                  |
| [basic_string<char_type>] | _M_capacity                |
| _M_dataplus               | _M_refcount                |
| _M_p ----->               | unnamed array of char_type |

Where the `_M_p` points to the first character in the string, and you cast it to a pointer-to-`_Rep` and subtract 1 to get a pointer to the header.

This approach has the enormous advantage that a string object requires only one allocation. All the ugliness is confined within a single pair of inline functions, which each compile to a single *add* instruction: `_Rep::_M_data()`, and `string::_M_rep()`; and the allocation function which gets a block of raw bytes and with room enough and constructs a `_Rep` object at the front.

The reason you want `_M_data` pointing to the character array and not the `_Rep` is so that the debugger can see the string contents. (Probably we should add a non-inline member to get the `_Rep` for the debugger to use, so users can check the actual string length.)

Note that the `_Rep` object is a POD so that you can have a static *empty string* `_Rep` object already *constructed* before static constructors have run. The reference-count encoding is chosen so that a 0 indicates one reference, so you never try to destroy the empty-string `_Rep` object.

All but the last paragraph is considered pretty conventional for a C++ string implementation.

Definition at line 107 of file `basic_string.h`.

## 5.390.2 Constructor & Destructor Documentation

**5.390.2.1** `template<typename _CharT, typename _Traits, typename _Alloc>  
std::basic_string<_CharT, _Traits, _Alloc>::basic_string ( )  
[inline]`

Default constructor creates an empty string.

Definition at line 432 of file `basic_string.h`.

Referenced by `std::basic_string<char>::substr()`.

**5.390.2.2** `template<typename _CharT, typename _Traits, typename _Alloc>  
std::basic_string<_CharT, _Traits, _Alloc>::basic_string ( const  
_Alloc & __a ) [explicit]`

### 5.390 **std::basic\_string<\_CharT, \_Traits, \_Alloc>** Class Template Reference

---

Construct an empty string using allocator *a*.

Definition at line 180 of file basic\_string.tcc.

**5.390.2.3** `template<typename _CharT, typename _Traits, typename _Alloc>  
std::basic_string<_CharT, _Traits, _Alloc>::basic_string ( const  
basic_string<_CharT, _Traits, _Alloc> & __str )`

Construct string with copy of value of *str*.

#### Parameters

*str* Source string.

Definition at line 172 of file basic\_string.tcc.

**5.390.2.4** `template<typename _CharT, typename _Traits, typename _Alloc>  
std::basic_string<_CharT, _Traits, _Alloc>::basic_string ( const  
basic_string<_CharT, _Traits, _Alloc> & __str, size_type __pos,  
size_type __n = npos )`

Construct string as copy of a substring.

#### Parameters

*str* Source string.

*pos* Index of first character to copy from.

*n* Number of characters to copy (default remainder).

Definition at line 186 of file basic\_string.tcc.

**5.390.2.5** `template<typename _CharT, typename _Traits, typename _Alloc>  
std::basic_string<_CharT, _Traits, _Alloc>::basic_string ( const  
basic_string<_CharT, _Traits, _Alloc> & __str, size_type __pos,  
size_type __n, const _Alloc & __a )`

Construct string as copy of a substring.

#### Parameters

*str* Source string.

## 5.390 `std::basic_string<_CharT, _Traits, _Alloc>` Class Template Reference 2204

---

*pos* Index of first character to copy from.

*n* Number of characters to copy.

*a* Allocator to use.

Definition at line 196 of file `basic_string.tcc`.

**5.390.2.6** `template<typename _CharT, typename _Traits, typename _Alloc>  
std::basic_string<_CharT, _Traits, _Alloc>::basic_string ( const  
_CharT * __s, size_type __n, const _Alloc & __a = _Alloc() )`

Construct string initialized by a character array.

### Parameters

*s* Source character array.

*n* Number of characters to copy.

*a* Allocator to use (default is default allocator).

NB: *s* must have at least *n* characters, `'\0'` has no special meaning.

Definition at line 208 of file `basic_string.tcc`.

**5.390.2.7** `template<typename _CharT, typename _Traits, typename _Alloc>  
std::basic_string<_CharT, _Traits, _Alloc>::basic_string ( const  
_CharT * __s, const _Alloc & __a = _Alloc() )`

Construct string as copy of a C string.

### Parameters

*s* Source C string.

*a* Allocator to use (default is default allocator).

Definition at line 215 of file `basic_string.tcc`.

**5.390.2.8** `template<typename _CharT, typename _Traits, typename _Alloc>  
std::basic_string<_CharT, _Traits, _Alloc>::basic_string (  
size_type __n, _CharT __c, const _Alloc & __a = _Alloc() )`

Construct string as multiple characters.

## 5.390 `std::basic_string<_CharT, _Traits, _Alloc>` Class Template Reference 2205

### Parameters

- n* Number of characters.
- c* Character to use.
- a* Allocator to use (default is default allocator).

Definition at line 222 of file `basic_string.tcc`.

**5.390.2.9** `template<typename _CharT, typename _Traits, typename _Alloc>  
std::basic_string<_CharT, _Traits, _Alloc>::basic_string (  
basic_string<_CharT, _Traits, _Alloc> && __str ) [inline]`

Move construct string.

### Parameters

- str* Source string.

The newly-created string contains the exact contents of *str*. *str* is a valid, but unspecified string.

Definition at line 502 of file `basic_string.h`.

**5.390.2.10** `template<typename _CharT, typename _Traits, typename _Alloc>  
std::basic_string<_CharT, _Traits, _Alloc>::basic_string (  
initializer_list<_CharT> __l, const _Alloc & __a = _Alloc() )`

Construct string from an initializer list.

### Parameters

- l* `std::initializer_list` of characters.
- a* Allocator to use (default is default allocator).

Definition at line 237 of file `basic_string.tcc`.

**5.390.2.11** `template<typename _CharT, typename _Traits, typename  
_Alloc> template<typename _InputIterator> std::basic_string<  
_CharT, _Traits, _Alloc>::basic_string ( _InputIterator __beg,  
_InputIterator __end, const _Alloc & __a = _Alloc() )`



## 5.390 `std::basic_string<_CharT, _Traits, _Alloc>` Class Template Reference 2206

---

Construct string as copy of a range.

### Parameters

- beg* Start of range.
- end* End of range.
- a* Allocator to use (default is default allocator).

Definition at line 230 of file `basic_string.tcc`.

```
5.390.2.12 template<typename _CharT, typename _Traits, typename _Alloc>
 std::basic_string< _CharT, _Traits, _Alloc >::~~basic_string ()
 [inline]
```

Destroy the string instance.

Definition at line 533 of file `basic_string.h`.

### 5.390.3 Member Function Documentation

```
5.390.3.1 template<typename _CharT , typename _Traits , typename _Alloc
 > basic_string< _CharT, _Traits, _Alloc > & std::basic_string<
 _CharT, _Traits, _Alloc >::append (const basic_string< _CharT,
 _Traits, _Alloc > & __str, size_type __pos, size_type __n)
```

Append a substring.

### Parameters

- str* The string to append.
- pos* Index of the first character of *str* to append.
- n* The number of characters to append.

### Returns

Reference to this string.

### Exceptions

[\*std::out\\_of\\_range\*](#) if *pos* is not a valid index.

## 5.390 `std::basic_string<_CharT, _Traits, _Alloc>` Class Template Reference 2207

---

This function appends  $n$  characters from *str* starting at *pos* to this string. If  $n$  is larger than the number of available characters in *str*, the remainder of *str* is appended.

Definition at line 344 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc>::capacity()`, `std::basic_string<_CharT, _Traits, _Alloc>::reserve()`, and `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

**5.390.3.2** `template<typename _CharT, typename _Traits, typename _Alloc  
> basic_string<_CharT, _Traits, _Alloc> & std::basic_string<  
_CharT, _Traits, _Alloc>::append ( const _CharT * __s, size_type  
__n )`

Append a C substring.

### Parameters

- $s$  The C string to append.
- $n$  The number of characters to append.

### Returns

Reference to this string.

Definition at line 300 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc>::capacity()`, `std::basic_string<_CharT, _Traits, _Alloc>::reserve()`, and `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

**5.390.3.3** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string<_CharT, _Traits, _Alloc>::append  
( const _CharT * __s ) [inline]`

Append a C string.

### Parameters

- $s$  The C string to append.

### Returns

Reference to this string.

Definition at line 997 of file `basic_string.h`.

**5.390.3.4** `template<typename _CharT, typename _Traits, typename _Alloc  
> basic_string<_CharT, _Traits, _Alloc> & std::basic_string<  
_CharT, _Traits, _Alloc>::append ( size_type __n, _CharT __c )`

Append multiple characters.

#### Parameters

*n* The number of characters to append.

*c* The character to use.

#### Returns

Reference to this string.

Appends *n* copies of *c* to this string.

Definition at line 283 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc>::capacity()`, `std::basic_string<_CharT, _Traits, _Alloc>::reserve()`, and `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

**5.390.3.5** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string<_CharT, _Traits, _Alloc>::append  
( initializer_list<_CharT> __l ) [inline]`

Append an [initializer\\_list](#) of characters.

#### Parameters

*l* The [initializer\\_list](#) of characters to append.

#### Returns

Reference to this string.

Definition at line 1021 of file `basic_string.h`.

Referenced by `std::basic_string<char>::append()`.

**5.390.3.6** `template<typename _CharT, typename _Traits, typename _Alloc>  
template<class _InputIterator> basic_string& std::basic_string<  
_CharT, _Traits, _Alloc>::append ( _InputIterator __first,  
_InputIterator __last ) [inline]`

Append a range of characters.

#### Parameters

*first* Iterator referencing the first character to append.

*last* Iterator marking the end of the range.

#### Returns

Reference to this string.

Appends characters in the range [first,last) to this string.

Definition at line 1035 of file `basic_string.h`.

**5.390.3.7** `template<typename _CharT, typename _Traits, typename _Alloc  
> basic_string<_CharT, _Traits, _Alloc> & std::basic_string<  
_CharT, _Traits, _Alloc>::append ( const basic_string<_CharT,  
_Traits, _Alloc> & __str )`

Append a string to this string.

#### Parameters

*str* The string to append.

#### Returns

Reference to this string.

Definition at line 327 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc>::capacity()`, `std::basic_string<_CharT, _Traits, _Alloc>::reserve()`, and `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

Referenced by `std::basic_string<char>::append()`, `std::collate<_CharT>::do_transform()`, `std::operator+()`, `std::basic_string<char>::operator+=()`, `std::operator>>()`, and `std::basic_string<_CharT, _Traits, _Alloc>::resize()`.

**5.390.3.8** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string<_CharT, _Traits, _Alloc>::assign (  
basic_string<_CharT, _Traits, _Alloc> && __str ) [inline]`

Set value to contents of another string.

#### Parameters

*str* Source string to use.

#### Returns

Reference to this string.

This function sets this string to the exact contents of *str*. *str* is a valid, but unspecified string.

Definition at line 1070 of file `basic_string.h`.

**5.390.3.9** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string<_CharT, _Traits, _Alloc>::assign (  
const basic_string<_CharT, _Traits, _Alloc> & __str, size_type  
__pos, size_type __n ) [inline]`

Set value to a substring of a string.

#### Parameters

*str* The string to use.

*pos* Index of the first character of *str*.

*n* Number of characters to use.

#### Returns

Reference to this string.

#### Exceptions

[\*std::out\\_of\\_range\*](#) if *pos* is not a valid index.

This function sets this string to the substring of *str* consisting of *n* characters at *pos*. If *n* is larger than the number of available characters in *str*, the remainder of *str* is used.

Definition at line 1090 of file `basic_string.h`.

Referenced by `std::basic_string<char>::assign()`.

**5.390.3.10** `template<typename _CharT, typename _Traits, typename _Alloc  
> basic_string<_CharT, _Traits, _Alloc> & std::basic_string<  
_CharT, _Traits, _Alloc>::assign ( const _CharT * __s, size_type  
__n )`

Set value to a C substring.

#### Parameters

- s* The C string to use.
- n* Number of characters to use.

#### Returns

Reference to this string.

This function sets the value of this string to the first *n* characters of *s*. If *n* is larger than the number of available characters in *s*, the remainder of *s* is used.

Definition at line 261 of file basic\_string.tcc.

References std::basic\_string<\_CharT, \_Traits, \_Alloc>::size().

**5.390.3.11** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string<_CharT, _Traits, _Alloc>::assign  
( const _CharT * __s ) [inline]`

Set value to contents of a C string.

#### Parameters

- s* The C string to use.

#### Returns

Reference to this string.

This function sets the value of this string to the value of *s*. The data is copied, so there is no dependence on *s* once the function returns.

Definition at line 1118 of file basic\_string.h.

**5.390.3.12** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string<_CharT, _Traits, _Alloc>::assign  
( size_type __n, _CharT __c ) [inline]`

Set value to multiple characters.

#### Parameters

- n* Length of the resulting string.
- c* The character to use.

#### Returns

Reference to this string.

This function sets the value of this string to *n* copies of character *c*.

Definition at line 1134 of file `basic_string.h`.

**5.390.3.13** `template<typename _CharT, typename _Traits, typename _Alloc>  
template<class _InputIterator> basic_string& std::basic_string<  
_CharT, _Traits, _Alloc>::assign ( _InputIterator __first,  
_InputIterator __last ) [inline]`

Set value to a range of characters.

#### Parameters

- first* Iterator referencing the first character to append.
- last* Iterator marking the end of the range.

#### Returns

Reference to this string.

Sets value of string to characters in the range [first,last).

Definition at line 1147 of file `basic_string.h`.

**5.390.3.14** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string<_CharT, _Traits, _Alloc>::assign  
( initializer_list<_CharT> __l ) [inline]`

### 5.390 std::basic\_string<\_CharT, \_Traits, \_Alloc> Class Template Reference 2213

Set value to an [initializer\\_list](#) of characters.

#### Parameters

*l* The [initializer\\_list](#) of characters to assign.

#### Returns

Reference to this string.

Definition at line 1157 of file basic\_string.h.

Referenced by std::basic\_string<char>::assign().

```
5.390.3.15 template<typename _CharT, typename _Traits, typename _Alloc
> basic_string< _CharT, _Traits, _Alloc > & std::basic_string<
_CharT, _Traits, _Alloc >::assign (const basic_string< _CharT,
_Traits, _Alloc > & __str)
```

Set value to contents of another string.

#### Parameters

*str* Source string to use.

#### Returns

Reference to this string.

Definition at line 245 of file basic\_string.tcc.

References std::basic\_string<\_CharT, \_Traits, \_Alloc>::get\_allocator().

Referenced by std::basic\_string<char>::assign(), std::basic\_string<char>::operator=(), std::basic\_stringbuf<\_CharT, \_Traits, \_Alloc>::overflow(), std::basic\_string<char>::push\_back(), and std::basic\_stringbuf<\_CharT, \_Traits, \_Alloc>::str().

```
5.390.3.16 template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string< _CharT, _Traits, _Alloc >::at (
size_type __n) const [inline]
```

Provides access to the data contained in the string.



### Parameters

*n* The index of the character to access.

### Returns

Read-only (const) reference to the character.

### Exceptions

*std::out\_of\_range* If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws *out\_of\_range* if the check fails.

Definition at line 856 of file `basic_string.h`.

**5.390.3.17** `template<typename _CharT, typename _Traits, typename _Alloc>  
reference std::basic_string<_CharT, _Traits, _Alloc>::at (  
size_type __n ) [inline]`

Provides access to the data contained in the string.

### Parameters

*n* The index of the character to access.

### Returns

Read/write reference to the character.

### Exceptions

*std::out\_of\_range* If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws *out\_of\_range* if the check fails. Success results in unsharing the string.

Definition at line 909 of file `basic_string.h`.

**5.390.3.18** `template<typename _CharT, typename _Traits, typename _Alloc>  
reference std::basic_string<_CharT, _Traits, _Alloc>::back ( )  
[inline]`

Returns a read/write reference to the data at the last element of the string.

Definition at line 885 of file `basic_string.h`.

## 5.390 std::basic\_string<\_CharT, \_Traits, \_Alloc> Class Template Reference 2215

---

**5.390.3.19** `template<typename _CharT, typename _Traits, typename _Alloc>  
const_reference std::basic_string<_CharT, _Traits, _Alloc>::back  
( ) const [inline]`

Returns a read-only (constant) reference to the data at the last element of the string.  
Definition at line 893 of file basic\_string.h.

**5.390.3.20** `template<typename _CharT, typename _Traits, typename _Alloc>  
iterator std::basic_string<_CharT, _Traits, _Alloc>::begin ( )  
[inline]`

Returns a read/write iterator that points to the first character in the string. Unshares the string.

Definition at line 600 of file basic\_string.h.

Referenced by std::basic\_string<char>::crend(), std::regex\_match(), std::regex\_replace(), std::regex\_search(), and std::basic\_string<char>::rend().

**5.390.3.21** `template<typename _CharT, typename _Traits, typename _Alloc>  
const_iterator std::basic_string<_CharT, _Traits, _Alloc>::begin ( ) const [inline]`

Returns a read-only (constant) iterator that points to the first character in the string.  
Definition at line 611 of file basic\_string.h.

**5.390.3.22** `template<typename _CharT, typename _Traits, typename _Alloc>  
const _CharT* std::basic_string<_CharT, _Traits, _Alloc>::c_str ( ) const [inline]`

Return const pointer to null-terminated contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1766 of file basic\_string.h.

Referenced by std::collate<\_CharT>::do\_compare(), std::money\_get<\_CharT, \_InIter>::do\_get(), std::num\_get<\_CharT, \_InIter>::do\_get(), std::collate<\_CharT>::do\_transform(), and std::basic\_filebuf<char\_type, traits\_type>::open().

**5.390.3.23** `template<typename _CharT, typename _Traits, typename _Alloc>  
size_type std::basic_string<_CharT, _Traits, _Alloc>::capacity (  
) const [inline]`

Returns the total number of characters that the string can hold before needing to allocate more memory.

Definition at line 768 of file `basic_string.h`.

Referenced by `std::basic_string<_CharT, _Traits, _Alloc>::append()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_string<char>::push_back()`, and `std::basic_string<_CharT, _Traits, _Alloc>::reserve()`.

**5.390.3.24** `template<typename _CharT, typename _Traits, typename _Alloc>  
const_iterator std::basic_string<_CharT, _Traits, _Alloc>::cbegin (  
) const [inline]`

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 675 of file `basic_string.h`.

**5.390.3.25** `template<typename _CharT, typename _Traits, typename _Alloc>  
const_iterator std::basic_string<_CharT, _Traits, _Alloc>::cend (  
) const [inline]`

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 683 of file `basic_string.h`.

**5.390.3.26** `template<typename _CharT, typename _Traits, typename _Alloc>  
void std::basic_string<_CharT, _Traits, _Alloc>::clear ( )  
[inline]`

Erases the string, making it empty.

Definition at line 795 of file `basic_string.h`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::setbuf()`.

**5.390.3.27** `template<typename _CharT, typename _Traits, typename _Alloc  
> int std::basic_string<_CharT, _Traits, _Alloc>::compare (  
size_type __pos, size_type __n1, const _CharT* __s, size_type  
__n2 ) const`

Compare substring against a character array.

#### Parameters

- pos1* Index of first character of substring.
- n1* Number of characters in substring.
- s* character array to compare against.
- n2* Number of characters of *s*.

#### Returns

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Form the substring of this string from the *n1* characters starting at *pos1*. Form a string from the first *n2* characters of *s*. Returns an integer  $< 0$  if this substring is ordered before the string from *s*,  $0$  if their values are equivalent, or  $> 0$  if this substring is ordered after the string from *s*. Determines the effective length *rlen* of the strings to compare as the smallest of the length of the substring and *n2*. The function then compares the two strings by calling `traits::compare(substring.data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

NB: *s* must have at least *n2* characters, `'\0'` has no special meaning.

Definition at line 982 of file `basic_string.tcc`.

References `std::min()`.

**5.390.3.28** `template<typename _CharT, typename _Traits, typename _Alloc  
> int std::basic_string<_CharT, _Traits, _Alloc>::compare (  
size_type __pos, size_type __n, const basic_string<_CharT,  
_Traits, _Alloc> & __str ) const`

Compare substring to a string.

#### Parameters

- pos* Index of first character of substring.
- n* Number of characters in substring.
- str* String to compare against.

#### Returns

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Form the substring of this string from the  $n$  characters starting at  $pos$ . Returns an integer  $< 0$  if the substring is ordered before  $str$ ,  $0$  if their values are equivalent, or  $> 0$  if the substring is ordered after  $str$ . Determines the effective length  $rlen$  of the strings to compare as the smallest of the length of the substring and  $str.size()$ . The function then compares the two strings by calling `traits::compare(substring.data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 918 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`, `std::basic_string<_CharT, _Traits, _Alloc>::data()`, `std::min()`, and `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

**5.390.3.29** `template<typename _CharT, typename _Traits, typename _Alloc  
> int std::basic_string<_CharT, _Traits, _Alloc>::compare (  
size_type __pos1, size_type __n1, const basic_string<_CharT,  
_Traits, _Alloc> & __str, size_type __pos2, size_type __n2 ) const`

Compare substring to a substring.

#### Parameters

- pos1* Index of first character of substring.
- n1* Number of characters in substring.
- str* String to compare against.
- pos2* Index of first character of substring of *str*.
- n2* Number of characters in substring of *str*.

#### Returns

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Form the substring of this string from the  $n1$  characters starting at  $pos1$ . Form the substring of *str* from the  $n2$  characters starting at  $pos2$ . Returns an integer  $< 0$  if this substring is ordered before the substring of *str*,  $0$  if their values are equivalent, or  $> 0$  if this substring is ordered after the substring of *str*. Determines the effective length  $rlen$  of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling `traits::compare(substring.data(), str.substr(pos2, n2).data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 933 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`, `std::basic_string<_CharT, _Traits, _Alloc>::data()`, and `std::min()`.

**5.390.3.30** `template<typename _CharT, typename _Traits, typename _Alloc>  
int std::basic_string<_CharT, _Traits, _Alloc>::compare (  
const basic_string<_CharT, _Traits, _Alloc> & __str ) const  
[inline]`

Compare to a string.

#### Parameters

*str* String to compare against.

#### Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before *str*, 0 if their values are equivalent, or > 0 if this string is ordered after *str*. Determines the effective length *rlen* of the strings to compare as the smallest of `size()` and `str.size()`. The function then compares the two strings by calling `traits::compare(data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 2173 of file `basic_string.h`.

Referenced by `std::sub_match<_Bi_iter>::compare()`, `std::basic_string<_CharT, _Traits, _Alloc>::compare()`, `std::basic_string<char>::compare()`, `std::operator<()`, `std::operator<=()`, `std::operator==()`, `std::operator>()`, and `std::operator>=()`.

**5.390.3.31** `template<typename _CharT, typename _Traits, typename _Alloc  
> int std::basic_string<_CharT, _Traits, _Alloc>::compare (  
size_type __pos, size_type __n1, const _CharT* __s ) const`

Compare substring to a C string.

#### Parameters

*pos* Index of first character of substring.

*n1* Number of characters in substring.

*s* C string to compare against.

#### Returns

Integer < 0, 0, or > 0.

### 5.390 std::basic\_string<\_CharT, \_Traits, \_Alloc> Class Template Reference

Form the substring of this string from the *nl* characters starting at *pos*. Returns an integer < 0 if the substring is ordered before *s*, 0 if their values are equivalent, or > 0 if the substring is ordered after *s*. Determines the effective length *rlen* of the strings to compare as the smallest of the length of the substring and the length of a string constructed from *s*. The function then compares the two string by calling `traits::compare(substring.data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 966 of file `basic_string.tcc`.

References `std::min()`.

**5.390.3.32** `template<typename _CharT, typename _Traits, typename _Alloc>  
int std::basic_string<_CharT, _Traits, _Alloc>::compare ( const  
_CharT * __s ) const`

Compare to a C string.

#### Parameters

*s* C string to compare against.

#### Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before *s*, 0 if their values are equivalent, or > 0 if this string is ordered after *s*. Determines the effective length *rlen* of the strings to compare as the smallest of `size()` and the length of a string constructed from *s*. The function then compares the two strings by calling `traits::compare(data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 951 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`, `std::basic_string<_CharT, _Traits, _Alloc>::length()`, `std::min()`, and `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

**5.390.3.33** `template<typename _CharT, typename _Traits, typename  
_Alloc> basic_string<_CharT, _Traits, _Alloc>::size_type  
std::basic_string<_CharT, _Traits, _Alloc>::copy ( _CharT * __s,  
size_type __n, size_type __pos = 0 ) const`

Copy substring into C string.

## 5.390 `std::basic_string<_CharT, _Traits, _Alloc>` Class Template Reference 2221

---

### Parameters

- s* C string to copy value into.
- n* Number of characters to copy.
- pos* Index of first character to copy.

### Returns

Number of characters actually copied

### Exceptions

*std::out\_of\_range* If *pos* > *size()*.

Copies up to *n* characters starting at *pos* into the C string *s*. If *pos* is greater than *size()*, *out\_of\_range* is thrown.

Definition at line 725 of file `basic_string.tcc`.

**5.390.3.34** `template<typename _CharT, typename _Traits, typename _Alloc>  
const_reverse_iterator std::basic_string<_CharT, _Traits, _Alloc  
>::crbegin ( ) const [inline]`

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 692 of file `basic_string.h`.

**5.390.3.35** `template<typename _CharT, typename _Traits, typename _Alloc>  
const_reverse_iterator std::basic_string<_CharT, _Traits, _Alloc  
>::crend ( ) const [inline]`

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 701 of file `basic_string.h`.

**5.390.3.36** `template<typename _CharT, typename _Traits, typename _Alloc>  
const _CharT* std::basic_string<_CharT, _Traits, _Alloc>::data ( ) const [inline]`

Return const pointer to contents.

This is a handle to internal data. Do not modify or dire things may happen.



## 5.390 `std::basic_string<_CharT, _Traits, _Alloc>` Class Template Reference 2222

Definition at line 1776 of file `basic_string.h`.

Referenced by `std::basic_string<_CharT, _Traits, _Alloc>::compare()`, `std::basic_string<char>::compare()`, `std::collate<_CharT>::do_compare()`, `std::collate<_CharT>::do_transform()`, `std::basic_string<char>::find()`, `std::basic_string<char>::find_first_not_of()`, `std::basic_string<char>::find_first_of()`, `std::basic_string<char>::find_last_not_of()`, `std::basic_string<char>::find_last_of()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_string<char>::rfind()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::str()`, and `std::regex_traits<_CharT>::transform()`.

**5.390.3.37** `template<typename _CharT, typename _Traits, typename _Alloc>  
bool std::basic_string<_CharT, _Traits, _Alloc>::empty ( ) const  
[inline]`

Returns true if the string is empty. Equivalent to `*this == ""`.

Definition at line 803 of file `basic_string.h`.

Referenced by `std::operator>>()`.

**5.390.3.38** `template<typename _CharT, typename _Traits, typename _Alloc>  
iterator std::basic_string<_CharT, _Traits, _Alloc>::end ( )  
[inline]`

Returns a read/write iterator that points one past the last character in the string. Unshares the string.

Definition at line 619 of file `basic_string.h`.

Referenced by `std::basic_string<char>::crbegin()`, `std::basic_string<char>::rbegin()`, `std::regex_match()`, `std::regex_replace()`, and `std::regex_search()`.

**5.390.3.39** `template<typename _CharT, typename _Traits, typename _Alloc>  
const_iterator std::basic_string<_CharT, _Traits, _Alloc>::end ( ) const  
[inline]`

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 630 of file `basic_string.h`.

**5.390.3.40** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string<_CharT, _Traits, _Alloc>::erase (  
size_type __pos = 0, size_type __n = npos ) [inline]`

Remove characters.

#### Parameters

*pos* Index of first character to remove (default 0).

*n* Number of characters to remove (default remainder).

#### Returns

Reference to this string.

#### Exceptions

[\*std::out\\_of\\_range\*](#) If *pos* is beyond the end of this string.

Removes *n* characters from this string starting at *pos*. The length of the string is reduced by *n*. If there are < *n* characters to remove, the remainder of the string is truncated. If *p* is beyond end of string, [\*out\\_of\\_range\*](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1347 of file `basic_string.h`.

Referenced by `std::getline()`, `std::operator>>()`, and `std::basic_string<_CharT, _Traits, _Alloc>::resize()`.

**5.390.3.41** `template<typename _CharT, typename _Traits, typename _Alloc>  
iterator std::basic_string<_CharT, _Traits, _Alloc>::erase (  
iterator __position ) [inline]`

Remove one character.

#### Parameters

*position* Iterator referencing the character to remove.

#### Returns

iterator referencing same location after removal.

Removes the character at *position* from this string. The value of the string doesn't change if an error is thrown.

Definition at line 1363 of file `basic_string.h`.

**5.390.3.42** `template<typename _CharT, typename _Traits, typename  
_Alloc> basic_string<_CharT, _Traits, _Alloc>::iterator  
std::basic_string<_CharT, _Traits, _Alloc>::erase ( iterator  
__first, iterator __last )`

Remove a range of characters.

#### Parameters

*first* Iterator referencing the first character to remove.

*last* Iterator referencing the end of the range.

#### Returns

Iterator referencing location of first after removal.

Removes the characters in the range [first,last) from this string. The value of the string doesn't change if an error is thrown.

Definition at line 393 of file `basic_string.tcc`.

**5.390.3.43** `template<typename _CharT, typename _Traits, typename  
_Alloc> basic_string<_CharT, _Traits, _Alloc>::size_type  
std::basic_string<_CharT, _Traits, _Alloc>::find ( const _CharT *  
__s, size_type __pos, size_type __n ) const`

Find position of a C substring.

#### Parameters

*s* C string to locate.

*pos* Index of character to search from.

*n* Number of characters from *s* to search for.

#### Returns

Index of start of first occurrence.

Starting from *pos*, searches forward for the first *n* characters in *s* within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 739 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc>::npos`, and `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

## 5.390 `std::basic_string<_CharT, _Traits, _Alloc>` Class Template Reference 2225

---

Referenced by `std::basic_string<_CharT, _Traits, _Alloc>::find()`, `std::basic_string<char>::find()`, `std::basic_string<_CharT, _Traits, _Alloc>::find_first_of()`, and `std::basic_string<char>::find_first_of()`.

**5.390.3.44** `template<typename _CharT, typename _Traits, typename _Alloc>  
size_type std::basic_string<_CharT, _Traits, _Alloc>::find ( const  
basic_string<_CharT, _Traits, _Alloc> & __str, size_type __pos =  
0 ) const [inline]`

Find position of a string.

### Parameters

*str* String to locate.

*pos* Index of character to search from (default 0).

### Returns

Index of start of first occurrence.

Starting from *pos*, searches forward for value of *str* within this string. If found, returns the index where it begins. If not found, returns *npos*.

Definition at line 1811 of file `basic_string.h`.

Referenced by `std::basic_string<char>::find()`.

**5.390.3.45** `template<typename _CharT, typename _Traits, typename _Alloc>  
size_type std::basic_string<_CharT, _Traits, _Alloc>::find ( const  
_CharT * __s, size_type __pos = 0 ) const [inline]`

Find position of a C string.

### Parameters

*s* C string to locate.

*pos* Index of character to search from (default 0).

### Returns

Index of start of first occurrence.

Starting from *pos*, searches forward for the value of *s* within this string. If found, returns the index where it begins. If not found, returns *npos*.

Definition at line 1825 of file `basic_string.h`.

**5.390.3.46** `template<typename _CharT, typename _Traits, typename  
_Alloc> basic_string<_CharT, _Traits, _Alloc>::size_type  
std::basic_string<_CharT, _Traits, _Alloc>::find ( _CharT __c,  
size_type __pos = 0 ) const`

Find position of a character.

#### Parameters

- c* Character to locate.
- pos* Index of character to search from (default 0).

#### Returns

Index of first occurrence.

Starting from *pos*, searches forward for *c* within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 762 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc>::find()`, `std::basic_string<_CharT, _Traits, _Alloc>::npos`, and `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

**5.390.3.47** `template<typename _CharT, typename _Traits, typename  
_Alloc> basic_string<_CharT, _Traits, _Alloc>::size_type  
std::basic_string<_CharT, _Traits, _Alloc>::find_first_not_of (   
_CharT __c, size_type __pos = 0 ) const`

Find position of a different character.

#### Parameters

- c* Character to avoid.
- pos* Index of character to search from (default 0).

#### Returns

Index of first occurrence.

Starting from *pos*, searches forward for a character other than *c* within this string. If found, returns the index where it was found. If not found, returns `npos`.

## 5.390 `std::basic_string<_CharT, _Traits, _Alloc>` Class Template Reference 2227

---

Definition at line 866 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc>::npos`, and `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

**5.390.3.48** `template<typename _CharT, typename _Traits, typename  
_Alloc> size_type std::basic_string<_CharT, _Traits, _Alloc  
>::find_first_not_of ( const basic_string<_CharT, _Traits, _Alloc  
> & __str, size_type __pos = 0 ) const [inline]`

Find position of a character not in string.

### Parameters

*str* String containing characters to avoid.

*pos* Index of character to search from (default 0).

### Returns

Index of first occurrence.

Starting from *pos*, searches forward for a character not contained in *str* within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 2035 of file `basic_string.h`.

Referenced by `std::basic_string<char>::find_first_not_of()`.

**5.390.3.49** `template<typename _CharT, typename _Traits, typename  
_Alloc> basic_string<_CharT, _Traits, _Alloc>::size_type  
std::basic_string<_CharT, _Traits, _Alloc>::find_first_not_of (   
const _CharT * __s, size_type __pos, size_type __n ) const`

Find position of a character not in C substring.

### Parameters

*s* C string containing characters to avoid.

*pos* Index of character to search from.

*n* Number of characters from *s* to consider.

### Returns

Index of first occurrence.

## 5.390 `std::basic_string<_CharT, _Traits, _Alloc>` Class Template Reference 2228

---

Starting from *pos*, searches forward for a character not contained in the first *n* characters of *s* within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 854 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc>::npos`, and `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

```
5.390.3.50 template<typename _CharT, typename _Traits, typename
 _Alloc> size_type std::basic_string<_CharT, _Traits, _Alloc
 >::find_first_not_of (const _CharT * __s, size_type __pos = 0)
 const [inline]
```

Find position of a character not in C string.

### Parameters

*s* C string containing characters to avoid.

*pos* Index of character to search from (default 0).

### Returns

Index of first occurrence.

Starting from *pos*, searches forward for a character not contained in *s* within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 2064 of file `basic_string.h`.

```
5.390.3.51 template<typename _CharT, typename _Traits, typename _Alloc>
 size_type std::basic_string<_CharT, _Traits, _Alloc>::find_first_of
 (const basic_string<_CharT, _Traits, _Alloc> & __str, size_type
 __pos = 0) const [inline]
```

Find position of a character of string.

### Parameters

*str* String containing characters to locate.

*pos* Index of character to search from (default 0).

### Returns

Index of first occurrence.

### 5.390 `std::basic_string<_CharT, _Traits, _Alloc>` Class Template Reference 2229

---

Starting from *pos*, searches forward for one of the characters of *str* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 1913 of file `basic_string.h`.

Referenced by `std::basic_string<char>::find_first_of()`.

**5.390.3.52** `template<typename _CharT, typename _Traits, typename  
_Alloc> basic_string<_CharT, _Traits, _Alloc>::size_type  
std::basic_string<_CharT, _Traits, _Alloc>::find_first_of( const  
_CharT* __s, size_type __pos, size_type __n ) const`

Find position of a character of C substring.

#### Parameters

- s* String containing characters to locate.
- pos* Index of character to search from.
- n* Number of characters from *s* to search for.

#### Returns

Index of first occurrence.

Starting from *pos*, searches forward for one of the first *n* characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 818 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc>::find()`, `std::basic_string<_CharT, _Traits, _Alloc>::npos`, and `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

**5.390.3.53** `template<typename _CharT, typename _Traits, typename _Alloc>  
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_first_of  
( const _CharT* __s, size_type __pos = 0 ) const [inline]`

Find position of a character of C string.

#### Parameters

- s* String containing characters to locate.
- pos* Index of character to search from (default 0).



### 5.390 `std::basic_string<_CharT, _Traits, _Alloc>` Class Template Reference 2230

---

#### Returns

Index of first occurrence.

Starting from *pos*, searches forward for one of the characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 1941 of file `basic_string.h`.

**5.390.3.54** `template<typename _CharT, typename _Traits, typename _Alloc>  
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_first_of  
( _CharT __c, size_type __pos = 0 ) const [inline]`

Find position of a character.

#### Parameters

*c* Character to locate.

*pos* Index of character to search from (default 0).

#### Returns

Index of first occurrence.

Starting from *pos*, searches forward for the character *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Note: equivalent to `find(c, pos)`.

Definition at line 1960 of file `basic_string.h`.

**5.390.3.55** `template<typename _CharT, typename _Traits, typename  
_Alloc> basic_string<_CharT, _Traits, _Alloc>::size_type  
std::basic_string<_CharT, _Traits, _Alloc>::find_last_not_of(  
const _CharT* __s, size_type __pos, size_type __n) const`

Find last position of a character not in C substring.

#### Parameters

*s* C string containing characters to avoid.

*pos* Index of character to search back from.

*n* Number of characters from *s* to consider.

### Returns

Index of last occurrence.

Starting from *pos*, searches backward for a character not contained in the first *n* characters of *s* within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 877 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc>::npos`, and `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

**5.390.3.56** `template<typename _CharT, typename _Traits, typename  
_Alloc> size_type std::basic_string<_CharT, _Traits, _Alloc  
>::find_last_not_of( const _CharT * __s, size_type __pos = npos )  
const [inline]`

Find last position of a character not in C string.

### Parameters

*s* C string containing characters to avoid.

*pos* Index of character to search back from (default end).

### Returns

Index of last occurrence.

Starting from *pos*, searches backward for a character not contained in *s* within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 2123 of file `basic_string.h`.

**5.390.3.57** `template<typename _CharT, typename _Traits, typename  
_Alloc> basic_string<_CharT, _Traits, _Alloc>::size_type  
std::basic_string<_CharT, _Traits, _Alloc>::find_last_not_of(  
_CharT __c, size_type __pos = npos ) const`

Find last position of a different character.

### Parameters

*c* Character to avoid.

*pos* Index of character to search back from (default end).

### Returns

Index of last occurrence.

Starting from *pos*, searches backward for a character other than *c* within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 898 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc>::npos`, and `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

**5.390.3.58** `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string<_CharT, _Traits, _Alloc>::find_last_not_of( const basic_string<_CharT, _Traits, _Alloc> & __str, size_type __pos = npos ) const [inline]`

Find last position of a character not in string.

### Parameters

*str* String containing characters to avoid.

*pos* Index of character to search back from (default end).

### Returns

Index of last occurrence.

Starting from *pos*, searches backward for a character not contained in *str* within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 2094 of file `basic_string.h`.

Referenced by `std::basic_string<char>::find_last_not_of()`.

**5.390.3.59** `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string<_CharT, _Traits, _Alloc>::find_last_of( _CharT __c, size_type __pos = npos ) const [inline]`

Find last position of a character.

### Parameters

*c* Character to locate.

## 5.390 std::basic\_string<\_CharT, \_Traits, \_Alloc> Class Template Reference 2233

---

*pos* Index of character to search back from (default end).

### Returns

Index of last occurrence.

Starting from *pos*, searches backward for *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Note: equivalent to `rfind(c, pos)`.

Definition at line 2021 of file `basic_string.h`.

```
5.390.3.60 template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_last_of
(const basic_string<_CharT, _Traits, _Alloc> & __str, size_type
__pos = npos) const [inline]
```

Find last position of a character of string.

### Parameters

*str* String containing characters to locate.

*pos* Index of character to search back from (default end).

### Returns

Index of last occurrence.

Starting from *pos*, searches backward for one of the characters of *str* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 1974 of file `basic_string.h`.

Referenced by `std::basic_string<char>::find_last_of()`.

```
5.390.3.61 template<typename _CharT, typename _Traits, typename
_Alloc> basic_string<_CharT, _Traits, _Alloc>::size_type
std::basic_string<_CharT, _Traits, _Alloc>::find_last_of(const
_CharT * __s, size_type __pos, size_type __n) const
```

Find last position of a character of C substring.

### Parameters

*s* C string containing characters to locate.

## 5.390 `std::basic_string<_CharT, _Traits, _Alloc>` Class Template Reference 2234

---

*pos* Index of character to search back from.

*n* Number of characters from *s* to search for.

### Returns

Index of last occurrence.

Starting from *pos*, searches backward for one of the first *n* characters of *s* within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 833 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc>::npos`, and `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

**5.390.3.62** `template<typename _CharT, typename _Traits, typename _Alloc>  
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_last_of  
( const _CharT * __s, size_type __pos = npos ) const [inline]`

Find last position of a character of C string.

### Parameters

*s* C string containing characters to locate.

*pos* Index of character to search back from (default end).

### Returns

Index of last occurrence.

Starting from *pos*, searches backward for one of the characters of *s* within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 2002 of file `basic_string.h`.

**5.390.3.63** `template<typename _CharT, typename _Traits, typename _Alloc>  
const_reference std::basic_string<_CharT, _Traits, _Alloc>::front  
( ) const [inline]`

Returns a read-only (constant) reference to the data at the first element of the string.

Definition at line 877 of file `basic_string.h`.

**5.390.3.64** `template<typename _CharT, typename _Traits, typename _Alloc>  
reference std::basic_string<_CharT, _Traits, _Alloc>::front ( )  
[inline]`

Returns a read/write reference to the data at the first element of the string.

Definition at line 869 of file `basic_string.h`.

**5.390.3.65** `template<typename _CharT, typename _Traits, typename _Alloc>  
allocator_type std::basic_string<_CharT, _Traits, _Alloc>::get_allocator ( ) const [inline]`

Return copy of allocator used to construct this string.

Definition at line 1783 of file `basic_string.h`.

Referenced by `std::basic_string<_CharT, _Traits, _Alloc>::assign()`, `std::basic_string<char>::basic_string()`, `std::basic_string<_CharT, _Traits, _Alloc>::reserve()`, `std::basic_string<_CharT, _Traits, _Alloc>::swap()`, and `std::basic_string<char>::~~basic_string()`.

**5.390.3.66** `template<typename _CharT, typename _Traits, typename _Alloc>  
> basic_string<_CharT, _Traits, _Alloc> & std::basic_string<  
_CharT, _Traits, _Alloc>::insert ( size_type __pos, const _CharT  
* __s, size_type __n )`

Insert a C substring.

### Parameters

*pos* Iterator referencing location in string to insert at.

*s* The C string to insert.

*n* The number of characters to insert.

### Returns

Reference to this string.

### Exceptions

[\*std::length\\_error\*](#) If new length exceeds `max_size()`.

[\*std::out\\_of\\_range\*](#) If *pos* is beyond the end of this string.

### 5.390 std::basic\_string<\_CharT, \_Traits, \_Alloc> Class Template Reference 2236

Inserts the first  $n$  characters of  $s$  starting at  $pos$ . If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If  $pos$  is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 362 of file `basic_string.tcc`.

```
5.390.3.67 template<typename _CharT, typename _Traits, typename _Alloc>
 iterator std::basic_string< _CharT, _Traits, _Alloc >::insert (
 iterator __p, _CharT __c) [inline]
```

Insert one character.

#### Parameters

- $p$  Iterator referencing position in string to insert at.
- $c$  The character to insert.

#### Returns

Iterator referencing newly inserted char.

#### Exceptions

`std::length_error` If new length exceeds `max_size()`.

Inserts character  $c$  at position referenced by  $p$ . If adding character causes the length to exceed `max_size()`, `length_error` is thrown. If  $p$  is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1323 of file `basic_string.h`.

```
5.390.3.68 template<typename _CharT, typename _Traits, typename _Alloc>
 void std::basic_string< _CharT, _Traits, _Alloc >::insert (iterator
 __p, size_type __n, _CharT __c) [inline]
```

Insert multiple characters.

#### Parameters

- $p$  Iterator referencing location in string to insert at.
- $n$  Number of characters to insert
- $c$  The character to insert.

### Exceptions

*`std::length_error`* If new length exceeds `max_size()`.

Inserts  $n$  copies of character  $c$  starting at the position referenced by iterator  $p$ . If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1174 of file `basic_string.h`.

Referenced by `std::basic_string<char>::insert()`.

```
5.390.3.69 template<typename _CharT, typename _Traits, typename _Alloc>
 basic_string& std::basic_string<_CharT, _Traits, _Alloc>::insert (
 size_type __pos1, const basic_string<_CharT, _Traits, _Alloc> &
 __str) [inline]
```

Insert value of a string.

### Parameters

*`pos1`* Iterator referencing location in string to insert at.

*`str`* The string to insert.

### Returns

Reference to this string.

### Exceptions

*`std::length_error`* If new length exceeds `max_size()`.

Inserts value of *`str`* starting at *`pos1`*. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1220 of file `basic_string.h`.

Referenced by `std::basic_string<char>::insert()`.

```
5.390.3.70 template<typename _CharT, typename _Traits, typename _Alloc>
 void std::basic_string<_CharT, _Traits, _Alloc>::insert (iterator
 __p, initializer_list<_CharT> __l) [inline]
```

Insert an `initializer_list` of characters.



**Parameters**

- p* Iterator referencing location in string to insert at.  
*l* The `initializer_list` of characters to insert.

**Exceptions**

`std::length_error` If new length exceeds `max_size()`.

Definition at line 1201 of file `basic_string.h`.

```
5.390.3.71 template<typename _CharT, typename _Traits, typename _Alloc>
 template<class _InputIterator > void std::basic_string< _CharT,
 _Traits, _Alloc >::insert (iterator __p, _InputIterator __beg,
 _InputIterator __end) [inline]
```

Insert a range of characters.

**Parameters**

- p* Iterator referencing location in string to insert at.  
*beg* Start of range.  
*end* End of range.

**Exceptions**

`std::length_error` If new length exceeds `max_size()`.

Inserts characters in range `[beg,end)`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1190 of file `basic_string.h`.

```
5.390.3.72 template<typename _CharT, typename _Traits, typename _Alloc>
 basic_string& std::basic_string< _CharT, _Traits, _Alloc >::insert (
 size_type __pos, size_type __n, _CharT __c) [inline]
```

Insert multiple characters.

**Parameters**

- pos* Index in string to insert at.

*n* Number of characters to insert

*c* The character to insert.

### Returns

Reference to this string.

### Exceptions

*std::length\_error* If new length exceeds `max_size()`.

*std::out\_of\_range* If *pos* is beyond the end of this string.

Inserts *n* copies of character *c* starting at index *pos*. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If *pos* > `length()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1306 of file `basic_string.h`.

```
5.390.3.73 template<typename _CharT, typename _Traits, typename _Alloc>
 basic_string& std::basic_string<_CharT, _Traits, _Alloc>::insert (
 size_type __pos1, const basic_string<_CharT, _Traits, _Alloc> &
 __str, size_type __pos2, size_type __n) [inline]
```

Insert a substring.

### Parameters

*pos1* Iterator referencing location in string to insert at.

*str* The string to insert.

*pos2* Start of characters in *str* to insert.

*n* Number of characters to insert.

### Returns

Reference to this string.

### Exceptions

*std::length\_error* If new length exceeds `max_size()`.

*std::out\_of\_range* If *pos1* > `size()` or *pos2* > *str.size()*.

Starting at *pos1*, insert *n* character of *str* beginning with *pos2*. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If *pos1* is beyond the

## 5.390 `std::basic_string<_CharT, _Traits, _Alloc>` Class Template Reference 2240

---

end of this string or *pos2* is beyond the end of *str*, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1242 of file `basic_string.h`.

Referenced by `std::basic_string<char>::insert()`.

**5.390.3.74** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string<_CharT, _Traits, _Alloc>::insert (  
size_type __pos, const _CharT* __s ) [inline]`

Insert a C string.

### Parameters

*pos* Iterator referencing location in string to insert at.

*s* The C string to insert.

### Returns

Reference to this string.

### Exceptions

`std::length_error` If new length exceeds `max_size()`.

`std::out_of_range` If *pos* is beyond the end of this string.

Inserts the first *n* characters of *s* starting at *pos*. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If *pos* is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1283 of file `basic_string.h`.

**5.390.3.75** `template<typename _CharT, typename _Traits, typename _Alloc>  
size_type std::basic_string<_CharT, _Traits, _Alloc>::length ( )  
const [inline]`

Returns the number of characters in the string, not including any /// null-termination.

Definition at line 716 of file `basic_string.h`.

Referenced by `std::basic_string<_CharT, _Traits, _Alloc>::compare()`, `std::collate<_CharT>::do_compare()`, `std::collate<_CharT>::do_transform()`, and `std::regex_traits<_Ch_type>::length()`.

**5.390.3.76** `template<typename _CharT, typename _Traits, typename _Alloc>  
size_type std::basic_string<_CharT, _Traits, _Alloc>::max_size (  
) const [inline]`

Returns the `size()` of the largest possible string.

Definition at line 721 of file `basic_string.h`.

Referenced by `std::getline()`, `std::operator>>()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`.

**5.390.3.77** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string<_CharT, _Traits, _Alloc>  
>::operator+=( const basic_string<_CharT, _Traits, _Alloc> &  
__str ) [inline]`

Append a string to this string.

#### Parameters

*str* The string to append.

#### Returns

Reference to this string.

Definition at line 924 of file `basic_string.h`.

**5.390.3.78** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string<_CharT, _Traits, _Alloc>  
>::operator+=( const _CharT * __s ) [inline]`

Append a C string.

#### Parameters

*s* The C string to append.

#### Returns

Reference to this string.

Definition at line 933 of file `basic_string.h`.

**5.390.3.79** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string<_CharT, _Traits, _Alloc  
>::operator+=( _CharT __c ) [inline]`

Append a character.

#### Parameters

*c* The character to append.

#### Returns

Reference to this string.

Definition at line 942 of file `basic_string.h`.

**5.390.3.80** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string<_CharT, _Traits, _Alloc  
>::operator+=( initializer_list<_CharT> __l ) [inline]`

Append an [initializer\\_list](#) of characters.

#### Parameters

*l* The [initializer\\_list](#) of characters to be appended.

#### Returns

Reference to this string.

Definition at line 955 of file `basic_string.h`.

**5.390.3.81** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string<_CharT, _Traits, _Alloc  
>::operator=( basic_string<_CharT, _Traits, _Alloc> && __str  
) [inline]`

Move assign the value of *str* to this string.

#### Parameters

*str* Source string.

## 5.390 `std::basic_string<_CharT, _Traits, _Alloc>` Class Template Reference 2243

---

The contents of *str* are moved into this string (without copying). *str* is a valid, but unspecified string.

Definition at line 575 of file `basic_string.h`.

**5.390.3.82** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string<_CharT, _Traits, _Alloc>  
>::operator= ( const basic_string<_CharT, _Traits, _Alloc> &  
__str ) [inline]`

Assign the value of *str* to this string.

### Parameters

*str* Source string.

Definition at line 541 of file `basic_string.h`.

**5.390.3.83** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string<_CharT, _Traits, _Alloc>  
>::operator= ( const _CharT * __s ) [inline]`

Copy contents of *s* into this string.

### Parameters

*s* Source null-terminated string.

Definition at line 549 of file `basic_string.h`.

**5.390.3.84** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string<_CharT, _Traits, _Alloc>  
>::operator= ( initializer_list<_CharT> __l ) [inline]`

Set value to string constructed from initializer list.

### Parameters

*l* [std::initializer\\_list](#).

Definition at line 587 of file `basic_string.h`.

**5.390.3.85** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string<_CharT, _Traits, _Alloc  
>::operator= ( _CharT __c ) [inline]`

Set value to string of length 1.

#### Parameters

*c* Source character.

Assigning to a character makes this string length 1 and (\*this)[0] == *c*.

Definition at line 560 of file basic\_string.h.

**5.390.3.86** `template<typename _CharT, typename _Traits, typename _Alloc>  
const_reference std::basic_string<_CharT, _Traits, _Alloc  
>::operator[] ( size_type __pos ) const [inline]`

Subscript access to the data contained in the string.

#### Parameters

*pos* The index of the character to access.

#### Returns

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and [out\\_of\\_range](#) lookups are not defined. (For checked lookups see [at\(\)](#).)

Definition at line 818 of file basic\_string.h.

Referenced by `std::basic_string< char >::back()`, and `std::basic_string< char >::front()`.

**5.390.3.87** `template<typename _CharT, typename _Traits, typename _Alloc>  
reference std::basic_string<_CharT, _Traits, _Alloc>::operator[] (   
size_type __pos ) [inline]`

Subscript access to the data contained in the string.

### Parameters

*pos* The index of the character to access.

### Returns

Read/write reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.) Unshares the string.

Definition at line 835 of file `basic_string.h`.

```
5.390.3.88 template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string<_CharT, _Traits, _Alloc>::push_back (
 _CharT __c) [inline]
```

Append a single character.

### Parameters

*c* Character to append.

Definition at line 1043 of file `basic_string.h`.

Referenced by `std::collate<_CharT>::do_transform()`, `std::basic_string<char>::operator+=()`, `std::operator>>()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`.

```
5.390.3.89 template<typename _CharT, typename _Traits, typename _Alloc>
reverse_iterator std::basic_string<_CharT, _Traits, _Alloc>
::rbegin () [inline]
```

Returns a read/write reverse iterator that points to the last character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 639 of file `basic_string.h`.

```
5.390.3.90 template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string<_CharT, _Traits, _Alloc>
::rbegin () const [inline]
```

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 648 of file `basic_string.h`.



**5.390.3.91** `template<typename _CharT, typename _Traits, typename _Alloc>  
const_reverse_iterator std::basic_string<_CharT, _Traits, _Alloc  
>::rend ( ) const [inline]`

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 666 of file `basic_string.h`.

**5.390.3.92** `template<typename _CharT, typename _Traits, typename _Alloc>  
reverse_iterator std::basic_string<_CharT, _Traits, _Alloc>::rend  
( ) [inline]`

Returns a read/write reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 657 of file `basic_string.h`.

**5.390.3.93** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace  
( size_type __pos, size_type __n, const basic_string<_CharT,  
_Traits, _Alloc> & __str ) [inline]`

Replace characters with value from another string.

#### Parameters

*pos* Index of first character to replace.  
*n* Number of characters to be replaced.  
*str* String to insert.

#### Returns

Reference to this string.

#### Exceptions

*[std::out\\_of\\_range](#)* If *pos* is beyond the end of this string.  
*[std::length\\_error](#)* If new length exceeds `max_size()`.

Removes the characters in the range `[pos, pos+n)` from this string. In place, the value of *str* is inserted. If *pos* is beyond end of string, *[out\\_of\\_range](#)* is thrown. If the length of the result exceeds `max_size()`, *[length\\_error](#)* is thrown. The value of the string doesn't change if an error is thrown.

## 5.390 std::basic\_string<\_CharT, \_Traits, \_Alloc> Class Template Reference 2247

---

Definition at line 1402 of file basic\_string.h.

Referenced by std::basic\_string< char >::append(), std::basic\_string< char >::assign(), std::basic\_string< char >::insert(), and std::basic\_string< char >::replace().

**5.390.3.94** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::replace  
( iterator __i1, iterator __i2, const basic_string< _CharT, _Traits,  
_Alloc > & __str ) [inline]`

Replace range of characters with string.

### Parameters

*i1* Iterator referencing start of range to replace.

*i2* Iterator referencing end of range to replace.

*str* String value to insert.

### Returns

Reference to this string.

### Exceptions

[\*std::length\\_error\*](#) If new length exceeds `max_size()`.

Removes the characters in the range [i1,i2). In place, the value of *str* is inserted. If the length of result exceeds `max_size()`, [\*length\\_error\*](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1508 of file basic\_string.h.

Referenced by std::basic\_string< char >::replace().

**5.390.3.95** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::replace  
( iterator __i1, iterator __i2, const _CharT * __s, size_type __n  
) [inline]`

Replace range of characters with C substring.

### Parameters

*i1* Iterator referencing start of range to replace.

*i2* Iterator referencing end of range to replace.

*s* C string value to insert.

*n* Number of characters from *s* to insert.

### Returns

Reference to this string.

### Exceptions

*std::length\_error* If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, the first *n* characters of *s* are inserted. If the length of result exceeds `max_size()`, *length\_error* is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1526 of file `basic_string.h`.

```
5.390.3.96 template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace
(iterator __i1, iterator __i2, const _CharT* __s) [inline]
```

Replace range of characters with C string.

### Parameters

*i1* Iterator referencing start of range to replace.

*i2* Iterator referencing end of range to replace.

*s* C string value to insert.

### Returns

Reference to this string.

### Exceptions

*std::length\_error* If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, the characters of *s* are inserted. If the length of result exceeds `max_size()`, *length\_error* is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1547 of file `basic_string.h`.

**5.390.3.97** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace  
( size_type __pos1, size_type __n1, const basic_string<_CharT,  
_Traits, _Alloc> & __str, size_type __pos2, size_type __n2 )  
[inline]`

Replace characters with value from another string.

#### Parameters

*pos1* Index of first character to replace.  
*n1* Number of characters to be replaced.  
*str* String to insert.  
*pos2* Index of first character of str to use.  
*n2* Number of characters from str to use.

#### Returns

Reference to this string.

#### Exceptions

[\*std::out\\_of\\_range\*](#) If *pos1* > [size\(\)](#) or *pos2* > [str.size\(\)](#).  
[\*std::length\\_error\*](#) If new length exceeds [max\\_size\(\)](#).

Removes the characters in the range [*pos1*, *pos1* + *n*) from this string. In place, the value of *str* is inserted. If *pos* is beyond end of string, [out\\_of\\_range](#) is thrown. If the length of the result exceeds [max\\_size\(\)](#), [length\\_error](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1424 of file `basic_string.h`.

Referenced by `std::basic_string<char>::replace()`.

**5.390.3.98** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace  
( size_type __pos, size_type __n1, size_type __n2, _CharT __c )  
[inline]`

Replace characters with multiple characters.

#### Parameters

*pos* Index of first character to replace.

*n1* Number of characters to be replaced.

*n2* Number of characters to insert.

*c* Character to insert.

#### Returns

Reference to this string.

#### Exceptions

[`std::out\_of\_range`](#) If *pos* > `size()`.

[`std::length\_error`](#) If new length exceeds `max_size()`.

Removes the characters in the range [*pos*, *pos* + *n1*) from this string. In place, *n2* copies of *c* are inserted. If *pos* is beyond end of string, [`out\_of\_range`](#) is thrown. If the length of result exceeds `max_size()`, [`length\_error`](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1490 of file `basic_string.h`.

```
5.390.3.99 template<typename _CharT, typename _Traits, typename _Alloc>
 basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace
 (iterator __i1, iterator __i2, size_type __n, _CharT __c)
 [inline]
```

Replace range of characters with multiple characters.

#### Parameters

*i1* Iterator referencing start of range to replace.

*i2* Iterator referencing end of range to replace.

*n* Number of characters to insert.

*c* Character to insert.

#### Returns

Reference to this string.

#### Exceptions

[`std::length\_error`](#) If new length exceeds `max_size()`.

Removes the characters in the range [*i1*, *i2*). In place, *n* copies of *c* are inserted. If the length of result exceeds `max_size()`, [`length\_error`](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1568 of file `basic_string.h`.

**5.390.3.100** `template<typename _CharT, typename _Traits, typename _Alloc>  
template<class _InputIterator > basic_string& std::basic_string<  
_CharT, _Traits, _Alloc >::replace ( iterator __i1, iterator __i2,  
_InputIterator __k1, _InputIterator __k2 ) [inline]`

Replace range of characters with range.

#### Parameters

- i1* Iterator referencing start of range to replace.
- i2* Iterator referencing end of range to replace.
- k1* Iterator referencing start of range to insert.
- k2* Iterator referencing end of range to insert.

#### Returns

Reference to this string.

#### Exceptions

*std::length\_error* If new length exceeds `max_size()`.

Removes the characters in the range [*i1*,*i2*). In place, characters in the range [*k1*,*k2*) are inserted. If the length of result exceeds `max_size()`, *length\_error* is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1591 of file `basic_string.h`.

**5.390.3.101** `template<typename _CharT, typename _Traits, typename _Alloc  
> basic_string<_CharT, _Traits, _Alloc > & std::basic_string<  
_CharT, _Traits, _Alloc >::replace ( size_type __pos, size_type  
__n1, const _CharT* __s, size_type __n2 )`

Replace characters with value of a C substring.

#### Parameters

- pos* Index of first character to replace.
- n1* Number of characters to be replaced.
- s* C string to insert.
- n2* Number of characters from *s* to use.

### Returns

Reference to this string.

### Exceptions

*`std::out_of_range`* If `pos1 > size()`.

*`std::length_error`* If new length exceeds `max_size()`.

Removes the characters in the range `[pos, pos + n1)` from this string. In place, the first `n2` characters of `s` are inserted, or all of `s` if `n2` is too large. If `pos` is beyond end of string, *`out_of_range`* is thrown. If the length of result exceeds `max_size()`, *`length_error`* is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 416 of file `basic_string.tcc`.

**5.390.3.102** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string<_CharT, _Traits, _Alloc>  
>::replace ( size_type __pos, size_type __n1, const _CharT* __s  
) [inline]`

Replace characters with value of a C string.

### Parameters

*`pos`* Index of first character to replace.

*`n1`* Number of characters to be replaced.

*`s`* C string to insert.

### Returns

Reference to this string.

### Exceptions

*`std::out_of_range`* If `pos > size()`.

*`std::length_error`* If new length exceeds `max_size()`.

Removes the characters in the range `[pos, pos + n1)` from this string. In place, the characters of `s` are inserted. If `pos` is beyond end of string, *`out_of_range`* is thrown. If the length of result exceeds `max_size()`, *`length_error`* is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1467 of file `basic_string.h`.

**5.390.3.103** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string& std::basic_string<_CharT, _Traits, _Alloc  
>::replace ( iterator __i1, iterator __i2, initializer_list<_CharT  
> __l ) [inline]`

Replace range of characters with [initializer\\_list](#).

#### Parameters

- i1* Iterator referencing start of range to replace.
- i2* Iterator referencing end of range to replace.
- l* The [initializer\\_list](#) of characters to insert.

#### Returns

Reference to this string.

#### Exceptions

[std::length\\_error](#) If new length exceeds `max_size()`.

Removes the characters in the range [i1,i2). In place, characters in the range [k1,k2) are inserted. If the length of result exceeds `max_size()`, [length\\_error](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1659 of file `basic_string.h`.

Referenced by `std::basic_string<char>::replace()`.

**5.390.3.104** `template<typename _CharT, typename _Traits, typename _Alloc  
> void std::basic_string<_CharT, _Traits, _Alloc>::reserve (   
size_type __res_arg = 0 )`

Attempt to preallocate enough memory for specified number of characters.

#### Parameters

- res\_arg* Number of characters required.

#### Exceptions

[std::length\\_error](#) If *res\_arg* exceeds `max_size()`.



This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

Definition at line 504 of file `basic_string.tcc`.

References `std::basic_string< _CharT, _Traits, _Alloc >::capacity()`, `std::basic_string< _CharT, _Traits, _Alloc >::get_allocator()`, and `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

Referenced by `std::basic_string< _CharT, _Traits, _Alloc >::append()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::operator>>()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_string< char >::push_back()`, and `std::basic_string< char >::shrink_to_fit()`.

**5.390.3.105** `template<typename _CharT, typename _Traits, typename _Alloc  
> void std::basic_string< _CharT, _Traits, _Alloc >::resize (`  
`size_type __n, _CharT __c )`

Resizes the string to the specified number of characters.

#### Parameters

- n* Number of characters the string should contain.
- c* Character to fill any new elements.

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to *c*.

Definition at line 642 of file `basic_string.tcc`.

References `std::basic_string< _CharT, _Traits, _Alloc >::append()`, `std::basic_string< _CharT, _Traits, _Alloc >::erase()`, and `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

Referenced by `std::money_get< _CharT, _InIter >::do_get()`.

**5.390.3.106** `template<typename _CharT, typename _Traits, typename _Alloc>`  
`void std::basic_string< _CharT, _Traits, _Alloc >::resize (`  
`size_type __n ) [inline]`

Resizes the string to the specified number of characters.

#### Parameters

*n* Number of characters the string should contain.

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as `char`, this means setting them to 0.

Definition at line 748 of file `basic_string.h`.

Referenced by `std::basic_string<char>::resize()`.

**5.390.3.107** `template<typename _CharT, typename _Traits, typename _Alloc>  
size_type std::basic_string<_CharT, _Traits, _Alloc>::rfind (  
const basic_string<_CharT, _Traits, _Alloc> & __str, size_type  
__pos = npos ) const [inline]`

Find last position of a string.

#### Parameters

*str* String to locate.

*pos* Index of character to search back from (default end).

#### Returns

Index of start of last occurrence.

Starting from *pos*, searches backward for value of *str* within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1855 of file `basic_string.h`.

Referenced by `std::basic_string<char>::find_last_of()`, and `std::basic_string<char>::rfind()`.

**5.390.3.108** `template<typename _CharT, typename _Traits, typename  
_Alloc> basic_string<_CharT, _Traits, _Alloc>::size_type  
std::basic_string<_CharT, _Traits, _Alloc>::rfind ( _CharT __c,  
size_type __pos = npos ) const`

Find last position of a character.

### Parameters

*c* Character to locate.

*pos* Index of character to search back from (default end).

### Returns

Index of last occurrence.

Starting from *pos*, searches backward for *c* within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 801 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc>::npos`, and `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

**5.390.3.109** `template<typename _CharT, typename _Traits, typename _Alloc>  
size_type std::basic_string<_CharT, _Traits, _Alloc>::rfind (  
const _CharT * __s, size_type __pos = npos ) const [inline]`

Find last position of a C string.

### Parameters

*s* C string to locate.

*pos* Index of character to start search at (default end).

### Returns

Index of start of last occurrence.

Starting from *pos*, searches backward for the value of *s* within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1883 of file `basic_string.h`.

**5.390.3.110** `template<typename _CharT, typename _Traits, typename  
_Alloc> basic_string<_CharT, _Traits, _Alloc>::size_type  
std::basic_string<_CharT, _Traits, _Alloc>::rfind ( const  
_CharT * __s, size_type __pos, size_type __n ) const`

Find last position of a C substring.

**Parameters**

- s* C string to locate.
- pos* Index of character to search back from.
- n* Number of characters from *s* to search for.

**Returns**

Index of start of last occurrence.

Starting from *pos*, searches backward for the first *n* characters in *s* within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 780 of file `basic_string.tcc`.

References `std::min()`, `std::basic_string<_CharT, _Traits, _Alloc>::npos`, and `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

**5.390.3.111** `template<typename _CharT, typename _Traits, typename _Alloc>`  
`void std::basic_string<_CharT, _Traits, _Alloc>::shrink_to_fit (`  
`) [inline]`

A non-binding request to reduce `capacity()` to `size()`.

Definition at line 754 of file `basic_string.h`.

**5.390.3.112** `template<typename _CharT, typename _Traits, typename _Alloc>`  
`size_type std::basic_string<_CharT, _Traits, _Alloc>::size ( )`  
`const [inline]`

Returns the number of characters in the string, not including any `///` null-termination.

Definition at line 710 of file `basic_string.h`.

Referenced by `std::basic_string<_CharT, _Traits, _Alloc>::append()`, `std::basic_string<_CharT, _Traits, _Alloc>::assign()`, `std::basic_string<char>::assign()`, `std::basic_string<char>::at()`, `std::basic_string<char>::back()`, `std::bitset<_S_match_flag_last>::bitset()`, `std::basic_string<char>::cend()`, `std::basic_string<char>::clear()`, `std::basic_string<_CharT, _Traits, _Alloc>::compare()`, `std::basic_string<char>::compare()`, `std::basic_string<char>::empty()`, `std::basic_string<char>::end()`, `std::basic_string<_CharT, _Traits, _Alloc>::find()`, `std::basic_string<char>::find()`, `std::basic_string<_CharT, _Traits, _Alloc>::find_first_not_of()`, `std::basic_string<char>::find_first_not_of()`, `std::basic_string<_CharT, _Traits, _Alloc>::find_first_of()`, `std::basic_string<char>::find_first_of()`, `std::basic_string<`

\_CharT, \_Traits, \_Alloc >::find\_last\_not\_of(), std::basic\_string< char >::find\_last\_not\_of(), std::basic\_string< \_CharT, \_Traits, \_Alloc >::find\_last\_of(), std::basic\_string< char >::find\_last\_of(), std::basic\_string< char >::insert(), std::operator+(), std::basic\_string< char >::operator[], std::basic\_string< char >::push\_back(), std::basic\_string< char >::replace(), std::basic\_string< \_CharT, \_Traits, \_Alloc >::reserve(), std::basic\_string< \_CharT, \_Traits, \_Alloc >::resize(), std::basic\_string< \_CharT, \_Traits, \_Alloc >::rfind(), std::basic\_string< char >::rfind(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::str(), and std::regex\_traits< \_CharT, \_Traits, \_Alloc >::transform().

**5.390.3.113** `template<typename _CharT, typename _Traits, typename _Alloc>  
basic_string std::basic_string< _CharT, _Traits, _Alloc >::substr (  
size_type __pos = 0, size_type __n = npos ) const [inline]`

Get a substring.

#### Parameters

- pos* Index of first character (default 0).
- n* Number of characters in substring (default remainder).

#### Returns

The new string.

#### Exceptions

[\*std::out\\_of\\_range\*](#) If *pos* > [\*size\(\)\*](#).

Construct and return a new string using the *n* characters starting at *pos*. If the string is too short, use the remainder of the characters. If *pos* is beyond the end of the string, [\*out\\_of\\_range\*](#) is thrown.

Definition at line 2155 of file basic\_string.h.

**5.390.3.114** `template<typename _CharT, typename _Traits, typename _Alloc>  
> void std::basic_string< _CharT, _Traits, _Alloc >::swap (  
basic_string< _CharT, _Traits, _Alloc > & __s )`

Swap contents with another string.

#### Parameters

- s* String to swap with.

Exchanges the contents of this string with that of *s* in constant time.

Definition at line 521 of file basic\_string.tcc.

References std::basic\_string< \_CharT, \_Traits, \_Alloc >::get\_allocator().

Referenced by std::basic\_string< char >::assign(), std::basic\_string< char >::operator=(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::overflow(), and std::swap().

#### **5.390.4 Member Data Documentation**

##### **5.390.4.1 template<typename \_CharT, typename \_Traits, typename \_Alloc> const basic\_string< \_CharT, \_Traits, \_Alloc >::size\_type std::basic\_string< \_CharT, \_Traits, \_Alloc >::npos [static]**

Value returned by various member functions when they fail.

Definition at line 280 of file basic\_string.h.

Referenced by std::basic\_string< \_CharT, \_Traits, \_Alloc >::find(), std::basic\_string< \_CharT, \_Traits, \_Alloc >::find\_first\_not\_of(), std::basic\_string< \_CharT, \_Traits, \_Alloc >::find\_first\_of(), std::basic\_string< \_CharT, \_Traits, \_Alloc >::find\_last\_not\_of(), std::basic\_string< \_CharT, \_Traits, \_Alloc >::find\_last\_of(), and std::basic\_string< \_CharT, \_Traits, \_Alloc >::rfind().

The documentation for this class was generated from the following files:

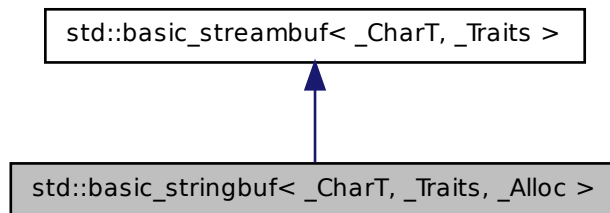
- [basic\\_string.h](#)
- [basic\\_string.tcc](#)

#### **5.391 std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc > Class Template Reference**

The actual work of input and output (for std::string).

This class associates either or both of its input and output sequences with a sequence of characters, which can be initialized from, or made available as, a [std::basic\\_string](#). (Paraphrased from [27.7.1]/1.).

Inheritance diagram for `std::basic_stringbuf<_CharT, _Traits, _Alloc>`:



### Public Types

- `typedef __string_type::size_type __size_type`
- `typedef basic_streambuf< char_type, traits_type > __streambuf_type`
- `typedef basic_string< char_type, _Traits, _Alloc > __string_type`
- `typedef _Alloc allocator_type`
- `typedef _CharT char_type`
- `typedef traits_type::int_type int_type`
- `typedef traits_type::off_type off_type`
- `typedef traits_type::pos_type pos_type`
- `typedef _Traits traits_type`

### Public Member Functions

- `basic_stringbuf (ios_base::openmode __mode=ios_base::in|ios_base::out)`
- `basic_stringbuf (const __string_type &__str, ios_base::openmode __mode=ios_base::in|ios_base::out)`
- `streamsize in_avail ()`
- `int_type sbumpc ()`
- `int_type sgetc ()`
- `streamsize sgetn (char_type *__s, streamsize __n)`
- `int_type snextc ()`
- `int_type sputbackc (char_type __c)`
- `int_type sputc (char_type __c)`
- `streamsize sputn (const char_type *__s, streamsize __n)`
- `__string_type str () const`

- `void str` (const `__string_type` &\_\_s)
- `int_type sungetc` ()

#### Protected Member Functions

- `void _M_stringbuf_init` (`ios_base::openmode` \_\_mode)
  - `void _M_sync` (`char_type` \*\_\_base, `__size_type` \_\_i, `__size_type` \_\_o)
  - `void _M_update_egptr` ()
  - `void gbump` (int \_\_n)
  - virtual `void imbue` (const `locale` &)
  - virtual `int_type overflow` (`int_type` \_\_c=traits\_type::eof())
  - virtual `int_type pbackfail` (`int_type` \_\_c=traits\_type::eof())
  - `void pbump` (int \_\_n)
  - virtual `pos_type seekoff` (`off_type` \_\_off, `ios_base::seekdir` \_\_way, `ios_base::openmode` \_\_mode=`ios_base::in`|`ios_base::out`)
  - virtual `pos_type seekpos` (`pos_type` \_\_sp, `ios_base::openmode` \_\_mode=`ios_base::in`|`ios_base::out`)
  - virtual `__streambuf_type * setbuf` (`char_type` \*\_\_s, `streamsize` \_\_n)
  - `void setg` (`char_type` \*\_\_gbeg, `char_type` \*\_\_gnext, `char_type` \*\_\_gend)
  - `void setp` (`char_type` \*\_\_pbeg, `char_type` \*\_\_pend)
  - virtual `streamsize showmanyc` ()
  - virtual `int sync` ()
  - virtual `int_type uflow` ()
  - virtual `int_type underflow` ()
  - virtual `streamsize xsgetn` (`char_type` \*\_\_s, `streamsize` \_\_n)
  - virtual `streamsize xspn` (const `char_type` \*\_\_s, `streamsize` \_\_n)
- 
- `char_type * eback` () const
  - `char_type * gptr` () const
  - `char_type * egptr` () const
- 
- `char_type * pbase` () const
  - `char_type * pptr` () const
  - `char_type * eptr` () const

#### Protected Attributes

- `ios_base::openmode` `_M_mode`
- `__string_type` `_M_string`



## Friends

- `template<bool _IsMove, typename _CharT2 >`  
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, _CharT2 * >::__-`  
`type __copy_move_a2 (istreambuf_iterator< _CharT2 >, istreambuf_iterator<`  
`_CharT2 >, _CharT2 *)`
  - `streamsize __copy_streambufs_eof (__streambuf_type *, __streambuf_type *,`  
`bool &)`
  - `class basic_ios< char_type, traits_type >`
  - `class basic_istream< char_type, traits_type >`
  - `class basic_ostream< char_type, traits_type >`
  - `template<typename _CharT2 >`  
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, istreambuf_-`  
`iterator< _CharT2 > >::__type find (istreambuf_iterator< _CharT2 >,`  
`istreambuf_iterator< _CharT2 >, const _CharT2 &)`
  - `template<typename _CharT2, typename _Traits2, typename _Alloc >`  
`basic_istream< _CharT2, _Traits2 > & getline (basic_istream< _CharT2, _-`  
`Traits2 > &, basic_string< _CharT2, _Traits2, _Alloc > &, _CharT2)`
  - `class istreambuf_iterator< char_type, traits_type >`
  - `template<typename _CharT2, typename _Traits2, typename _Alloc >`  
`basic_istream< _CharT2, _Traits2 > & operator>> (basic_istream< _CharT2,`  
`_Traits2 > &, basic_string< _CharT2, _Traits2, _Alloc > &)`
  - `template<typename _CharT2, typename _Traits2 >`  
`basic_istream< _CharT2, _Traits2 > & operator>> (basic_istream< _CharT2,`  
`_Traits2 > &, _CharT2 *)`
  - `class ostreambuf_iterator< char_type, traits_type >`
- 
- `char_type * _M_in_beg`
  - `char_type * _M_in_cur`
  - `char_type * _M_in_end`
  - `char_type * _M_out_beg`
  - `char_type * _M_out_cur`
  - `char_type * _M_out_end`
  - `locale _M_buf_locale`
  - `locale pubimbue (const locale &__loc)`
  - `locale getloc () const`
  - `__streambuf_type * pubsetbuf (char_type *__s, streamsize __n)`
  - `pos_type pubseekoff (off_type __off, ios_base::seekdir __way, ios_-`  
`base::openmode __mode=ios_base::in|ios_base::out)`
  - `pos_type pubseekpos (pos_type __sp, ios_base::openmode __mode=ios_-`  
`base::in|ios_base::out)`
  - `int pubsync ()`

### 5.391.1 Detailed Description

**template<typename \_CharT, typename \_Traits, typename \_Alloc> class std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >**

The actual work of input and output (for std::string).

This class associates either or both of its input and output sequences with a sequence of characters, which can be initialized from, or made available as, a [std::basic\\_string](#). (Paraphrased from [27.7.1]/1.). For this class, open modes (of type [ios\\_base::openmode](#)) have `in` set if the input sequence can be read, and `out` set if the output sequence can be written.

Definition at line 60 of file sstream.

### 5.391.2 Member Typedef Documentation

**5.391.2.1 template<typename \_CharT, typename \_Traits, typename \_Alloc> typedef basic\_streambuf<char\_type, traits\_type> std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::\_\_streambuf\_type**

This is a non-standard type.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 73 of file sstream.

**5.391.2.2 template<typename \_CharT, typename \_Traits, typename \_Alloc> typedef \_CharT std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::char\_type**

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 64 of file sstream.

**5.391.2.3 template<typename \_CharT, typename \_Traits, typename \_Alloc> typedef traits\_type::int\_type std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::int\_type**

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which

are specific to the implementation.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 69 of file sstream.

**5.391.2.4** `template<typename _CharT, typename _Traits, typename _Alloc>  
typedef traits_type::off_type std::basic_stringbuf< _CharT, _Traits,  
_Alloc >::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 71 of file sstream.

**5.391.2.5** `template<typename _CharT, typename _Traits, typename _Alloc>  
typedef traits_type::pos_type std::basic_stringbuf< _CharT, _Traits,  
_Alloc >::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 70 of file sstream.

**5.391.2.6** `template<typename _CharT, typename _Traits, typename _Alloc>  
typedef _Traits std::basic_stringbuf< _CharT, _Traits, _Alloc  
>::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 65 of file sstream.

### 5.391.3 Constructor & Destructor Documentation

**5.391.3.1** `template<typename _CharT, typename _Traits, typename _Alloc>  
std::basic_stringbuf<_CharT, _Traits, _Alloc>::basic_stringbuf  
( ios_base::openmode __mode = ios_base::in | ios_base::out )  
[inline, explicit]`

Starts with an empty string buffer.

#### Parameters

*mode* Whether the buffer can read, or write, or both.

The default constructor initializes the parent class using its own default ctor.

Definition at line 94 of file sstream.

**5.391.3.2** `template<typename _CharT, typename _Traits, typename _Alloc>  
std::basic_stringbuf<_CharT, _Traits, _Alloc>::basic_stringbuf  
( const __string_type & __str, ios_base::openmode __mode =  
ios_base::in | ios_base::out ) [inline, explicit]`

Starts with an existing string buffer.

#### Parameters

*str* A string to copy as a starting buffer.

*mode* Whether the buffer can read, or write, or both.

This constructor initializes the parent class using its own default ctor.

Definition at line 107 of file sstream.

### 5.391.4 Member Function Documentation

**5.391.4.1** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf<_CharT, _Traits>::eback ( ) const  
[inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- [eback\(\)](#) returns the beginning pointer for the input sequence
- [gptr\(\)](#) returns the next pointer for the input sequence
- [egptr\(\)](#) returns the end pointer for the input sequence

Definition at line 462 of file streambuf.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_destroy_pback()`, `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_streambuf< char_type, traits_type >::sputbackc()`, `std::basic_streambuf< char_type, traits_type >::sungetc()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

**5.391.4.2 template<typename \_CharT, typename \_Traits> char\_type\*  
std::basic\_streambuf< \_CharT, \_Traits >::egptr ( ) const  
[inline, protected, inherited]**

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- [eback\(\)](#) returns the beginning pointer for the input sequence
- [gptr\(\)](#) returns the next pointer for the input sequence
- [egptr\(\)](#) returns the end pointer for the input sequence

Definition at line 468 of file streambuf.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_create_pback()`, `std::basic_streambuf< char_type, traits_type >::in_avail()`, `std::basic_streambuf< char_type, traits_type >::sbumpc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_streambuf< char_type, traits_type >::sgetc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::str()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, `std::basic_streambuf< _CharT, _Traits >::xsgetn()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

**5.391.4.3** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::pptr ( ) const  
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 515 of file streambuf.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_streambuf< char_type, traits_type >::sputc()`, `std::basic_streambuf< _CharT, _Traits >::xsputn()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

**5.391.4.4** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf< _CharT, _Traits >::gbump ( int __n )  
[inline, protected, inherited]`

Moving the read position.

#### Parameters

- n* The delta by which to move.

This just advances the read position without returning any data.

Definition at line 478 of file streambuf.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_streambuf< char_type, traits_type >::sbumpc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_streambuf< char_type, traits_type >::sputbackc()`, `std::basic_streambuf< char_type, traits_type >::sungetc()`, `std::basic_streambuf< char_type, traits_type >::uflow()`, `std::basic_streambuf< _CharT, _Traits >::xsgetn()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

**5.391.4.5** `template<typename _CharT, typename _Traits> locale  
std::basic_streambuf< _CharT, _Traits >::getloc ( ) const  
[inline, inherited]`

Locale access.

#### Returns

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 224 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::pubimbue()`.

**5.391.4.6** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::gptr ( ) const  
[inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 465 of file `streambuf`.

Referenced by `std::basic_filebuf< char_type, traits_type >::M_create_pback()`, `std::basic_filebuf< char_type, traits_type >::M_destroy_pback()`, `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_streambuf< char_type, traits_type >::in_avail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_streambuf< char_type, traits_type >::sbumpc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_streambuf< char_type, traits_type >::sgetc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_streambuf< char_type, traits_type >::sputbackc()`, `std::basic_streambuf< char_type, traits_type`

>::sungetc(), std::basic\_streambuf< char\_type, traits\_type >::uflow(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::underflow(), std::basic\_filebuf< \_CharT, \_Traits >::underflow(), std::basic\_streambuf< \_CharT, \_Traits >::xsgetn(), and std::basic\_filebuf< \_CharT, \_Traits >::xsgetn().

**5.391.4.7 template<typename \_CharT, typename \_Traits> virtual void  
std::basic\_streambuf< \_CharT, \_Traits >::imbue ( const locale & )  
[inline, protected, virtual, inherited]**

Changes translations.

#### Parameters

*loc* A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

#### Note

Base class version does nothing.

Reimplemented in [std::basic\\_filebuf< \\_CharT, \\_Traits >](#), [std::basic\\_filebuf< \\_CharT, encoding\\_char\\_traits< \\_CharT > >](#), and [std::basic\\_filebuf< char\\_type, traits\\_type >](#).

Definition at line 556 of file streambuf.

Referenced by [std::basic\\_streambuf< char\\_type, traits\\_type >::pubimbue\(\)](#).

**5.391.4.8 template<typename \_CharT, typename \_Traits> streamsize  
std::basic\_streambuf< \_CharT, \_Traits >::in\_avail ( ) [inline,  
inherited]**

Looking ahead into the stream.

#### Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived [showmanyc\(\)](#).

Definition at line 264 of file streambuf.



5.391.4.9 `template<class _CharT , class _Traits , class _Alloc >`  
`basic_stringbuf< _CharT, _Traits, _Alloc >::int_type`  
`std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow ( int_type`  
`= traits_type::eof() ) [protected, virtual]`

Consumes data from the buffer; writes to the controlled sequence.

#### Parameters

*c* An additional character to consume.

#### Returns

`eof()` to indicate failure, something else (usually *c*, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if *c* is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

#### Note

Base class version does nothing, returns `eof()`.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 82 of file `sstream.tcc`.

References `std::basic_stringbuf< _CharT, _Traits, _Alloc >::_M_mode`, `std::basic_string< _CharT, _Traits, _Alloc >::assign()`, `std::basic_string< _CharT, _Traits, _Alloc >::capacity()`, `std::basic_string< _CharT, _Traits, _Alloc >::data()`, `std::basic_streambuf< _CharT, _Traits >::eback()`, `std::basic_streambuf< _CharT, _Traits >::epptr()`, `std::basic_streambuf< _CharT, _Traits >::gptr()`, `std::max()`, `std::basic_string< _CharT, _Traits, _Alloc >::max_size()`, `std::min()`, `std::ios_base::out`, `std::basic_streambuf< _CharT, _Traits >::pbase()`, `std::basic_streambuf< _CharT, _Traits >::pbump()`, `std::basic_streambuf< _CharT, _Traits >::pptr()`, `std::basic_string< _CharT, _Traits, _Alloc >::push_back()`, `std::basic_string< _CharT, _Traits, _Alloc >::reserve()`, and `std::basic_string< _CharT, _Traits, _Alloc >::swap()`.

**5.391.4.10** `template<class _CharT , class _Traits , class _Alloc >  
basic_stringbuf< _CharT, _Traits, _Alloc >::int_type  
std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail (  
int_type = traits_type::eof() ) [protected, virtual]`

Tries to back up the input sequence.

#### Parameters

*c* The character to be inserted back into the sequence.

#### Returns

*eof()* on failure, *some other value* on success

#### Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

#### Note

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 48 of file `sstream.tcc`.

References `std::basic_stringbuf< _CharT, _Traits, _Alloc >::M_mode`, `std::basic_streambuf< _CharT, _Traits >::eback()`, `std::basic_streambuf< _CharT, _Traits >::gbump()`, `std::basic_streambuf< _CharT, _Traits >::gptr()`, and `std::ios_base::out`.

**5.391.4.11** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::pbase ( ) const  
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 509 of file streambuf.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_filebuf< _CharT, _Traits >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::str()`, `std::basic_filebuf< _CharT, _Traits >::sync()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

**5.391.4.12** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf< _CharT, _Traits >::pbump ( int __n )  
[inline, protected, inherited]`

Moving the write position.

#### Parameters

*n* The delta by which to move.

This just advances the write position without returning any data.

Definition at line 525 of file streambuf.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_streambuf< char_type, traits_type >::sputc()`, and `std::basic_streambuf< _CharT, _Traits >::xsputn()`.

**5.391.4.13** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::pptr ( ) const  
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 512 of file streambuf.

Referenced by std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::overflow(), std::basic\_filebuf< \_CharT, \_Traits >::overflow(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_filebuf< \_CharT, \_Traits >::seekoff(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos(), std::basic\_streambuf< char\_type, traits\_type >::putc(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::str(), std::basic\_filebuf< \_CharT, \_Traits >::sync(), std::basic\_streambuf< \_CharT, \_Traits >::xsputn(), and std::basic\_filebuf< \_CharT, \_Traits >::xsputn().

**5.391.4.14** `template<typename _CharT, typename _Traits> locale  
std::basic_streambuf< _CharT, _Traits >::pubimbue ( const locale  
& __loc ) [inline, inherited]`

Entry point for [imbue\(\)](#).

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived imbue(loc).

Definition at line 207 of file streambuf.

**5.391.4.15** `template<typename _CharT, typename _Traits> pos_type  
std::basic_streambuf< _CharT, _Traits >::pubseekoff ( off_type  
__off, ios_base::seekdir __way, ios_base::openmode __mode =  
ios_base::in | ios_base::out ) [inline, inherited]`

Entry point for [imbue\(\)](#).

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived imbue(loc).

Definition at line 241 of file streambuf.

**5.391.4.16** `template<typename _CharT, typename _Traits> pos_type  
std::basic_streambuf<_CharT, _Traits>::pubseekpos ( pos_type  
__sp, ios_base::openmode __mode = ios_base::in | ios_base::out )  
[inline, inherited]`

Entry point for [imbue\(\)](#).

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 246 of file `streambuf`.

**5.391.4.17** `template<typename _CharT, typename _Traits>  
__streambuf_type* std::basic_streambuf<_CharT, _Traits  
>::pubsetbuf ( char_type * __s, streamsize __n ) [inline,  
inherited]`

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 237 of file `streambuf`.

**5.391.4.18** `template<typename _CharT, typename _Traits> int  
std::basic_streambuf<_CharT, _Traits>::pubsync ( )  
[inline, inherited]`

Entry point for [imbue\(\)](#).

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived imbue(loc).

Definition at line 251 of file streambuf.

Referenced by std::basic\_istream< \_CharT, \_Traits >::sync().

**5.391.4.19** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf< _CharT, _Traits >::sbumpc( ) [inline,  
inherited]`

Getting the next character.

#### Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 296 of file streambuf.

Referenced by std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::istreambuf\_iterator< \_CharT, \_Traits >::operator++(), and std::basic\_streambuf< char\_type, traits\_type >::snextc().

**5.391.4.20** `template<class _CharT , class _Traits , class _Alloc >  
basic_stringbuf< _CharT, _Traits, _Alloc >::pos_type  
std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff  
( off_type , ios_base::seekdir , ios_base::openmode =  
ios_base::in | ios_base::out ) [protected, virtual]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

#### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 151 of file sstream.tcc.

References std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::M\_mode, std::ios\_base::cur, std::basic\_streambuf< \_CharT, \_Traits >::eback(), std::basic\_streambuf<

`_CharT, _Traits >::egptr()`, `std::ios_base::end`, `std::basic_streambuf< _CharT, _Traits >::gbump()`, `std::basic_streambuf< _CharT, _Traits >::gptr()`, `std::ios_base::in`, `std::ios_base::out`, `std::basic_streambuf< _CharT, _Traits >::pbase()`, `std::basic_streambuf< _CharT, _Traits >::pbump()`, and `std::basic_streambuf< _CharT, _Traits >::pptr()`.

**5.391.4.21** `template<class _CharT , class _Traits , class _Alloc >  
basic_stringbuf< _CharT, _Traits, _Alloc >::pos_type  
std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos (   
pos_type, ios_base::openmode = ios_base::in | ios_base::out )  
[protected, virtual]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

#### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 198 of file `sstream.tcc`.

References `std::basic_stringbuf< _CharT, _Traits, _Alloc >::M_mode`, `std::basic_streambuf< _CharT, _Traits >::eback()`, `std::basic_streambuf< _CharT, _Traits >::egptr()`, `std::basic_streambuf< _CharT, _Traits >::gbump()`, `std::basic_streambuf< _CharT, _Traits >::gptr()`, `std::ios_base::in`, `std::ios_base::out`, `std::basic_streambuf< _CharT, _Traits >::pbase()`, `std::basic_streambuf< _CharT, _Traits >::pbump()`, and `std::basic_streambuf< _CharT, _Traits >::pptr()`.

**5.391.4.22** `template<typename _CharT, typename _Traits, typename _Alloc>  
virtual __streambuf_type* std::basic_stringbuf< _CharT, _Traits,  
_Alloc >::setbuf ( char_type * __s, streamsize __n ) [inline,  
protected, virtual]`

Manipulates the buffer.

#### Parameters

*s* Pointer to a buffer area.

*n* Size of *s*.

## Returns

`this`

If no buffer has already been created, and both *s* and *n* are non-zero, then *s* is used as a buffer; see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 198 of file `sstream`.

References [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::clear\(\)](#).

**5.391.4.23** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf< _CharT, _Traits >::setg ( char_type *  
__gbeg, char_type * __gnext, char_type * __gend ) [inline,  
protected, inherited]`

Setting the three read area pointers.

## Parameters

*gbeg* A pointer.

*gnext* A pointer.

*gend* A pointer.

## Postcondition

*gbeg* == `eback()`, *gnext* == `gptr()`, and *gend* == `egptr()`

Definition at line 489 of file `streambuf`.

Referenced by [std::basic\\_filebuf< char\\_type, traits\\_type >::\\_M\\_create\\_pback\(\)](#), [std::basic\\_filebuf< char\\_type, traits\\_type >::\\_M\\_destroy\\_pback\(\)](#), and [std::basic\\_filebuf< char\\_type, traits\\_type >::\\_M\\_set\\_buffer\(\)](#).

**5.391.4.24** `template<typename _CharT, typename _Traits> void  
std::basic_streambuf< _CharT, _Traits >::setp ( char_type  
* __pbeg, char_type * __pend ) [inline, protected,  
inherited]`

Setting the three write area pointers.



**Parameters**

*pbeg* A pointer.

*pend* A pointer.

**Postcondition**

*pbeg* == `pbase()`, *pbeg* == `pptr()`, and *pend* == `epptr()`

Definition at line 535 of file streambuf.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_set_buffer()`.

**5.391.4.25** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf< _CharT, _Traits >::sgetc ( ) [inline,  
inherited]`

Getting the next character.

**Returns**

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 318 of file streambuf.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_streambuf< char_type, traits_type >::snextc()`.

**5.391.4.26** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf< _CharT, _Traits >::sgetn ( char_type * __s,  
streamsize __n ) [inline, inherited]`

Entry point for `xsgetn`.

**Parameters**

*s* A buffer area.

*n* A count.

Returns `xsgetn(s,n)`. The effect is to fill `s[0]` through `s[n-1]` with characters from the input sequence, if possible.

Definition at line 337 of file `streambuf`.

**5.391.4.27** `template<typename _CharT, typename _Traits, typename _Alloc>`  
`virtual streamsize std::basic_stringbuf< _CharT, _Traits, _Alloc`  
`>::showmanyc( ) [inline, protected, virtual]`

Investigating the data available.

#### Returns

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.*  
[27.5.2.4.3]/1

#### Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 166 of file `sstream`.

References `std::basic_stringbuf< _CharT, _Traits, _Alloc >::_M_mode`, `std::basic_streambuf< _CharT, _Traits >::egptr()`, `std::basic_streambuf< _CharT, _Traits >::gptr()`, and `std::ios_base::in`.

**5.391.4.28** `template<typename _CharT, typename _Traits> int_type`  
`std::basic_streambuf< _CharT, _Traits >::snextc( ) [inline,`  
`inherited]`

Getting the next character.

#### Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 278 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

**5.391.4.29** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf< _CharT, _Traits >::sputbackc ( char_type  
__c ) [inline, inherited]`

Pushing characters back into the input stream.

#### Parameters

*c* The character to push back.

#### Returns

The previous character, if possible.

Similar to `sungetc()`, but *c* is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be *c*.

Definition at line 352 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::putback()`.

**5.391.4.30** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf< _CharT, _Traits >::sputc ( char_type __c )  
[inline, inherited]`

Entry point for all single-character output functions.

#### Parameters

*c* A character to output.

#### Returns

*c*, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores *c* in that position, increments the position, and returns `traits::to_int_type(c)`. If a write position is not available, returns `overflow(c)`.

Definition at line 404 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, and `std::ostreambuf_iterator< _CharT, _Traits >::operator=()`.

**5.391.4.31** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf< _CharT, _Traits >::sputn ( const char_type  
* __s, streamsize __n ) [inline, inherited]`

Entry point for all single-character output functions.

**Parameters**

*s* A buffer read area.

*n* A count.

One of two public output functions.

Returns `xspn(s,n)`. The effect is to write `s[0]` through `s[n-1]` to the output sequence, if possible.

Definition at line 430 of file `streambuf`.

**5.391.4.32** `template<typename _CharT, typename _Traits, typename _Alloc>  
void std::basic_stringbuf< _CharT, _Traits, _Alloc >::str ( const  
__string_type & __s ) [inline]`

Setting a new buffer.

**Parameters**

*s* The string to use as a new sequence.

Deallocates any previous stored sequence, then copies *s* to use as a new one.

Definition at line 146 of file `sstream`.

References `std::basic_stringbuf< _CharT, _Traits, _Alloc >::_M_mode`, `std::basic_string< _CharT, _Traits, _Alloc >::assign()`, `std::basic_string< _CharT, _Traits, _Alloc >::data()`, and `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

**5.391.4.33** `template<typename _CharT, typename _Traits, typename _Alloc>  
__string_type std::basic_stringbuf< _CharT, _Traits, _Alloc >::str ( ) const [inline]`

Copying out the string buffer.

#### Returns

A copy of one of the underlying sequences.

*If the buffer is only created in input mode, the underlying character sequence is equal to the input sequence; otherwise, it is equal to the output sequence. [27.7.1.2]/1*

Definition at line 122 of file sstream.

References std::basic\_streambuf< \_CharT, \_Traits >::egptr(), std::basic\_streambuf< \_CharT, \_Traits >::pbase(), and std::basic\_streambuf< \_CharT, \_Traits >::pptr().

**5.391.4.34** `template<typename _CharT, typename _Traits> int_type  
std::basic_streambuf< _CharT, _Traits >::sungetc ( ) [inline,  
inherited]`

Moving backwards in the input stream.

#### Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns [pbackfail\(\)](#). The effect is to *unget* the last character *gotten*.

Definition at line 377 of file streambuf.

Referenced by std::basic\_istream< \_CharT, \_Traits >::unget().

**5.391.4.35** `template<typename _CharT, typename _Traits> virtual int  
std::basic_streambuf< _CharT, _Traits >::sync ( void )  
[inline, protected, virtual, inherited]`

Synchronizes the buffer arrays with the controlled sequences.

**Returns**

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

**Note**

Base class version does nothing, returns zero.

Reimplemented in [std::basic\\_filebuf< \\_CharT, \\_Traits >, \\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf< \\_CharT, \\_Traits >, std::basic\\_filebuf< \\_CharT, encoding\\_char\\_traits< \\_CharT > >](#), and [std::basic\\_filebuf< char\\_type, traits\\_type >](#).

Definition at line 607 of file streambuf.

Referenced by [std::basic\\_streambuf< char\\_type, traits\\_type >::pubsync\(\)](#).

**5.391.4.36** `template<typename _CharT, typename _Traits> virtual int_type  
std::basic_streambuf< _CharT, _Traits >::uflow ( ) [inline,  
protected, virtual, inherited]`

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function does the same thing as [underflow\(\)](#), and in fact is required to call that function. It also returns the new character, like [underflow\(\)](#) does. However, this function also moves the read position forward by one.

Reimplemented in [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf< \\_CharT, \\_Traits >](#).

Definition at line 680 of file streambuf.

Referenced by [std::basic\\_streambuf< char\\_type, traits\\_type >::sbumpc\(\)](#), and [std::basic\\_streambuf< \\_CharT, \\_Traits >::xsgetn\(\)](#).

**5.391.4.37** `template<class _CharT , class _Traits , class _Alloc >  
basic_stringbuf< _CharT, _Traits, _Alloc >::int_type  
std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow ( )  
[protected, virtual]`

Fetches more data from the controlled sequence.

## Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25.html>

## Note

Base class version does nothing, returns `eof()`.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 133 of file `sstream.tcc`.

References `std::basic_stringbuf< _CharT, _Traits, _Alloc >::_M_mode`, `std::basic_streambuf< _CharT, _Traits >::egptr()`, `std::basic_streambuf< _CharT, _Traits >::gptr()`, and `std::ios_base::in`.

**5.391.4.38** `template<typename _CharT, typename _Traits> streamsize  
std::basic_streambuf< _CharT, _Traits >::xsgetn ( char_type *  
__s, streamsize __n ) [protected, virtual, inherited]`

Multiple character extraction.

## Parameters

*s* A buffer area.

*n* Maximum number of characters to assign.

## Returns

The number of characters assigned.

Fills `s[0]` through `s[n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either *n* characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in [std::basic\\_filebuf< \\_CharT, \\_Traits >, \\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf< \\_CharT, \\_Traits >, std::basic\\_filebuf< \\_CharT, encoding\\_char\\_traits< \\_CharT > >](#), and [std::basic\\_filebuf< char\\_type, traits\\_type >](#).

Definition at line 47 of file streambuf.tcc.

References [std::basic\\_streambuf< \\_CharT, \\_Traits >::egptr\(\)](#), [std::basic\\_streambuf< \\_CharT, \\_Traits >::gbump\(\)](#), [std::basic\\_streambuf< \\_CharT, \\_Traits >::gpptr\(\)](#), [std::min\(\)](#), and [std::basic\\_streambuf< \\_CharT, \\_Traits >::uflow\(\)](#).

Referenced by [std::basic\\_streambuf< char\\_type, traits\\_type >::sgetn\(\)](#).

**5.391.4.39 template<typename \_CharT, typename \_Traits > streamsize  
std::basic\_streambuf< \_CharT, \_Traits >::xsputn ( const  
char\_type \* \_\_s, streamsize \_\_n ) [protected, virtual,  
inherited]**

Multiple character insertion.

#### Parameters

- s* A buffer area.
- n* Maximum number of characters to write.

#### Returns

The number of characters written.

Writes *s*[0] through *s*[*n*-1] to the output sequence, as if by [sputc\(\)](#). Stops when either *n* characters have been copied, or when [sputc\(\)](#) would return [traits::eof\(\)](#).

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in [std::basic\\_filebuf< \\_CharT, \\_Traits >, \\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf< \\_CharT, \\_Traits >, std::basic\\_filebuf< \\_CharT, encoding\\_char\\_traits< \\_CharT > >](#), and [std::basic\\_filebuf< char\\_type, traits\\_type >](#).

Definition at line 81 of file streambuf.tcc.

References [std::basic\\_streambuf< \\_CharT, \\_Traits >::epptr\(\)](#), [std::min\(\)](#), [std::basic\\_streambuf< \\_CharT, \\_Traits >::overflow\(\)](#), [std::basic\\_streambuf< \\_CharT, \\_Traits >::pbump\(\)](#), and [std::basic\\_streambuf< \\_CharT, \\_Traits >::pptr\(\)](#).

Referenced by [std::basic\\_streambuf< char\\_type, traits\\_type >::sputn\(\)](#).



### 5.391.5 Member Data Documentation

#### 5.391.5.1 `template<typename _CharT, typename _Traits> locale std::basic_streambuf< _CharT, _Traits >::_M_buf_locale [protected, inherited]`

Current locale setting.

Definition at line 190 of file `streambuf`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::basic_filebuf()`, `std::basic_streambuf< char_type, traits_type >::getloc()`, and `std::basic_streambuf< char_type, traits_type >::pubimbue()`.

#### 5.391.5.2 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_beg [protected, inherited]`

This is based on `_IO_FILE`, just reordered to be more consistent, and is intended to be the most minimal abstraction for an internal buffer.

- `get == input == read`
- `put == output == write`

Definition at line 182 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::eback()`, and `std::basic_streambuf< char_type, traits_type >::setg()`.

#### 5.391.5.3 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_cur [protected, inherited]`

Entry point for `imbue()`.

### Parameters

*loc* The new locale.

### Returns

The previous locale.

Calls the derived imbue(loc).

Definition at line 183 of file streambuf.

Referenced by std::basic\_streambuf< char\_type, traits\_type >::gbump(), std::basic\_streambuf< char\_type, traits\_type >::gptr(), and std::basic\_streambuf< char\_type, traits\_type >::setg().

**5.391.5.4 template<typename \_CharT, typename \_Traits> char\_type\*  
std::basic\_streambuf< \_CharT, \_Traits >::\_M\_in\_end  
[protected, inherited]**

Entry point for [imbue\(\)](#).

**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

Calls the derived imbue(loc).

Definition at line 184 of file streambuf.

Referenced by std::basic\_streambuf< char\_type, traits\_type >::egptr(), and std::basic\_streambuf< char\_type, traits\_type >::setg().

**5.391.5.5 template<typename \_CharT, typename \_Traits, typename \_Alloc>  
ios\_base::openmode std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc  
>::\_M\_mode [protected]**

Place to stash in || out || in | out settings for current stringbuf.

Definition at line 79 of file sstream.

Referenced by std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::overflow(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::pbackfail(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::showmanyc(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::str(), and std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::underflow().

**5.391.5.6** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_out_beg  
[protected, inherited]`

Entry point for [imbue\(\)](#).

**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 185 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::pbase()`, and `std::basic_streambuf< char_type, traits_type >::setp()`.

**5.391.5.7** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_out_cur  
[protected, inherited]`

Entry point for [imbue\(\)](#).

**Parameters**

*loc* The new locale.

**Returns**

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 186 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::pbump()`, `std::basic_streambuf< char_type, traits_type >::pptr()`, and `std::basic_streambuf< char_type, traits_type >::setp()`.

**5.391.5.8** `template<typename _CharT, typename _Traits> char_type*  
std::basic_streambuf< _CharT, _Traits >::_M_out_end  
[protected, inherited]`

Entry point for [imbue\(\)](#).

#### Parameters

*loc* The new locale.

#### Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 187 of file `streambuf`.

Referenced by `std::basic_streambuf< char_type, traits_type >::epptr()`, and `std::basic_streambuf< char_type, traits_type >::setp()`.

The documentation for this class was generated from the following files:

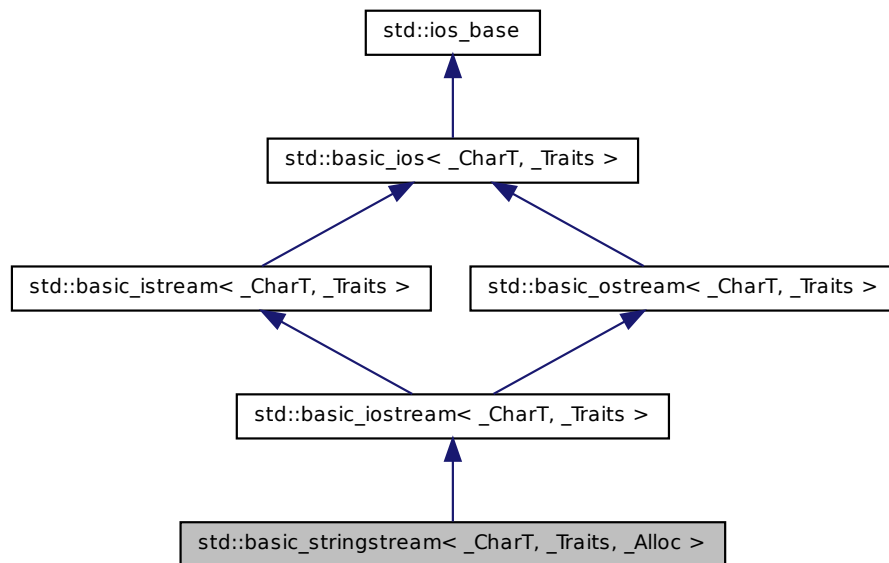
- [sstream](#)
- [sstream.tcc](#)

## **5.392** `std::basic_stringstream< _CharT, _Traits, _Alloc >` Class Template Reference

Controlling input and output for `std::string`.

This class supports reading from and writing to objects of type [std::basic\\_string](#), using the inherited functions from [std::basic\\_istream](#). To control the associated sequence, an instance of [std::basic\\_stringbuf](#) is used, which this page refers to as `sb`.

Inheritance diagram for `std::basic_stringstream<_CharT, _Traits, _Alloc>`:



### Public Types

- typedef `ctype<_CharT>` `__ctype_type`
- typedef `ctype<_CharT>` `__ctype_type`
- typedef `basic_ios<_CharT, _Traits>` `__ios_type`
- typedef `basic_ios<_CharT, _Traits>` `__ios_type`
- typedef `basic_iostream<char_type, traits_type>` `__iostream_type`
- typedef `basic_istream<_CharT, _Traits>` `__istream_type`
- typedef `basic_istream<_CharT, _Traits>` `__istream_type`
- typedef `num_get<_CharT, istreambuf_iterator<_CharT, _Traits>>` `__num_get_type`
- typedef `num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>>` `__num_put_type`
- typedef `basic_ostream<_CharT, _Traits>` `__ostream_type`
- typedef `basic_streambuf<_CharT, _Traits>` `__streambuf_type`
- typedef `basic_streambuf<_CharT, _Traits>` `__streambuf_type`
- typedef `basic_string<_CharT, _Traits, _Alloc>` `__string_type`

- typedef `basic_stringbuf<_CharT, _Traits, _Alloc>` `__stringbuf_type`
- typedef `_Alloc` `allocator_type`
- typedef `_CharT` `char_type`
- typedef `_CharT` `char_type`
- enum `event` { `erase_event`, `imbue_event`, `copyfmt_event` }
- typedef void(\* `event_callback`)(`event`, `ios_base` &, int)
- typedef `_Ios_Fmtflags` `fmtflags`
- typedef `_Traits::int_type` `int_type`
- typedef `traits_type::int_type` `int_type`
- typedef int `io_state`
- typedef `_Ios_Iostate` `iostate`
- typedef `traits_type::off_type` `off_type`
- typedef `_Traits::off_type` `off_type`
- typedef int `open_mode`
- typedef `_Ios_Openmode` `openmode`
- typedef `traits_type::pos_type` `pos_type`
- typedef `_Traits::pos_type` `pos_type`
- typedef int `seek_dir`
- typedef `_Ios_Seekdir` `seekdir`
- typedef `std::streamoff` `streamoff`
- typedef `std::streampos` `streampos`
- typedef `_Traits` `traits_type`
- typedef `_Traits` `traits_type`
  
- typedef `num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>>` `__num_put_type`

### Public Member Functions

- `basic_stringstream` (`ios_base::openmode` `__m=ios_base::out|ios_base::in`)
- `basic_stringstream` (const `__string_type` &`__str`, `ios_base::openmode` `__m=ios_base::out|ios_base::in`)
- `~basic_stringstream` ()
- const `locale` & `_M_getloc` () const
- void `_M_setstate` (`iostate` `__state`)
- bool `bad` () const
- void `clear` (`iostate` `__state=goodbit`)
- `basic_ios` & `copyfmt` (const `basic_ios` &`__rhs`)
- bool `eof` () const
- `iostate exceptions` () const
- void `exceptions` (`iostate` `__except`)
- bool `fail` () const

- `char_type fill () const`
- `char_type fill (char_type __ch)`
- `fmtflags flags () const`
- `fmtflags flags (fmtflags __fmtfl)`
- `__ostream_type & flush ()`
- `streamsize gcount () const`
- `template<>`  
`basic_istream< char > & getline (char_type *__s, streamsize __n, char_type`  
`__delim)`
- `template<>`  
`basic_istream< wchar_t > & getline (char_type *__s, streamsize __n, char_type`  
`__delim)`
- `locale getloc () const`
- `bool good () const`
- `template<>`  
`basic_istream< char > & ignore (streamsize __n, int_type __delim)`
- `template<>`  
`basic_istream< char > & ignore (streamsize __n)`
- `template<>`  
`basic_istream< wchar_t > & ignore (streamsize __n)`
- `template<>`  
`basic_istream< wchar_t > & ignore (streamsize __n, int_type __delim)`
- `locale imbue (const locale &__loc)`
- `long & iword (int __ix)`
- `char narrow (char_type __c, char __dfault) const`
- `streamsize precision () const`
- `streamsize precision (streamsize __prec)`
- `void *& pword (int __ix)`
- `__stringbuf_type * rdbuf () const`
- `basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits`  
`> *__sb)`
- `iostate rdstate () const`
- `void register_callback (event_callback __fn, int __index)`
- `__ostream_type & seekp (pos_type)`
- `__ostream_type & seekp (off_type, ios_base::seekdir)`
- `fmtflags setf (fmtflags __fmtfl)`
- `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
- `void setstate (iostate __state)`
- `__string_type str () const`
- `void str (const __string_type &__s)`
- `pos_type tellp ()`
- `basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > *__-`  
`_tiestr)`

- `basic_ostream<_CharT, _Traits> * tie ()` const
- void `unsetf (fmtflags __mask)`
- `char_type widen (char __c)` const
- `streamsize width (streamsize __wide)`
- `streamsize width ()` const
- `__istream_type & operator>> (__istream_type &(__pf)(__istream_type &))`
- `__istream_type & operator>> (__ios_type &(__pf)(__ios_type &))`
- `__istream_type & operator>> (ios_base &(__pf)(ios_base &))`

### Arithmetic Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__istream_type & operator>> (bool &__n)`
- `__istream_type & operator>> (short &__n)`
- `__istream_type & operator>> (unsigned short &__n)`
- `__istream_type & operator>> (int &__n)`
- `__istream_type & operator>> (unsigned int &__n)`
- `__istream_type & operator>> (long &__n)`
- `__istream_type & operator>> (unsigned long &__n)`
- `__istream_type & operator>> (long long &__n)`
- `__istream_type & operator>> (unsigned long long &__n)`
- `__istream_type & operator>> (float &__f)`
- `__istream_type & operator>> (double &__f)`
- `__istream_type & operator>> (long double &__f)`
- `__istream_type & operator>> (void *&__p)`
- `__istream_type & operator>> (__streambuf_type *__sb)`

### Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.



The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `int_type get ()`
- `__istream_type & get (char_type &__c)`
- `__istream_type & get (char_type * __s, streamsize __n, char_type __delim)`
- `__istream_type & get (char_type * __s, streamsize __n)`
- `__istream_type & get (__streambuf_type &__sb, char_type __delim)`
- `__istream_type & get (__streambuf_type &__sb)`
- `__istream_type & getline (char_type * __s, streamsize __n, char_type __delim)`
- `__istream_type & getline (char_type * __s, streamsize __n)`
- `__istream_type & ignore ()`
- `__istream_type & ignore (streamsize __n)`
- `__istream_type & ignore (streamsize __n, int_type __delim)`
- `int_type peek ()`
- `__istream_type & read (char_type * __s, streamsize __n)`
- `streamsize readsome (char_type * __s, streamsize __n)`
- `__istream_type & putback (char_type __c)`
- `__istream_type & unget ()`
- `int sync ()`
- `pos_type tellg ()`
- `__istream_type & seekg (pos_type)`
- `__istream_type & seekg (off_type, ios_base::seekdir)`
- `operator void * () const`
- `bool operator! () const`
- `__ostream_type & operator<< (__ostream_type &(__pf)(__ostream_type &))`
- `__ostream_type & operator<< (__ios_type &(__pf)(__ios_type &))`
- `__ostream_type & operator<< (ios_base &(__pf)(ios_base &))`

### Arithmetic Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__ostream_type & operator<< (long __n)`

- `__ostream_type & operator<< (unsigned long __n)`
- `__ostream_type & operator<< (bool __n)`
- `__ostream_type & operator<< (short __n)`
- `__ostream_type & operator<< (unsigned short __n)`
- `__ostream_type & operator<< (int __n)`
- `__ostream_type & operator<< (unsigned int __n)`
- `__ostream_type & operator<< (long long __n)`
- `__ostream_type & operator<< (unsigned long long __n)`
- `__ostream_type & operator<< (double __f)`
- `__ostream_type & operator<< (float __f)`
- `__ostream_type & operator<< (long double __f)`
- `__ostream_type & operator<< (const void *__p)`
- `__ostream_type & operator<< (__streambuf_type *__sb)`

### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If badbit is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type & put (char_type __c)`
- `void _M_write (const char_type *__s, streamsize __n)`
- `__ostream_type & write (const char_type *__s, streamsize __n)`

### Static Public Member Functions

- static bool `sync_with_stdio` (bool \_\_sync=true)
- static int `xalloc` () throw ()

### Static Public Attributes

- static const `fmtflags adjustfield`
- static const `openmode app`
- static const `openmode ate`
- static const `iostate badbit`
- static const `fmtflags basefield`
- static const `seekdir beg`
- static const `openmode binary`
- static const `fmtflags boolalpha`

- static const [seekdir](#) `cur`
- static const [fmtflags](#) `dec`
- static const [seekdir](#) `end`
- static const [iostate](#) `eofbit`
- static const [iostate](#) `failbit`
- static const [fmtflags](#) `fixed`
- static const [fmtflags](#) `floatfield`
- static const [iostate](#) `goodbit`
- static const [fmtflags](#) `hex`
- static const [openmode](#) `in`
- static const [fmtflags](#) `internal`
- static const [fmtflags](#) `left`
- static const [fmtflags](#) `oct`
- static const [openmode](#) `out`
- static const [fmtflags](#) `right`
- static const [fmtflags](#) `scientific`
- static const [fmtflags](#) `showbase`
- static const [fmtflags](#) `showpoint`
- static const [fmtflags](#) `showpos`
- static const [fmtflags](#) `skipws`
- static const [openmode](#) `trunc`
- static const [fmtflags](#) `unitbuf`
- static const [fmtflags](#) `uppercase`

### Protected Types

- enum { `_S_local_word_size` }

### Protected Member Functions

- void `_M_cache_locale` (const [locale](#) &\_\_loc)
- void `_M_call_callbacks` ([event](#) \_\_ev) throw ()
- void `_M_dispose_callbacks` (void) throw ()
- template<typename \_ValueT >  
  [\\_\\_istream\\_type](#) & `_M_extract` (\_ValueT &\_\_v)
- `_Words` & `_M_grow_words` (int \_\_index, bool \_\_iword)
- void `_M_init` () throw ()
- template<typename \_ValueT >  
  [\\_\\_ostream\\_type](#) & `_M_insert` (\_ValueT \_\_v)
- void `init` ([basic\\_streambuf](#)<\_CharT, \_Traits> \*\_\_sb)

### Protected Attributes

- `_Callback_list` \* `_M_callbacks`
- `const __ctype_type` \* `_M_ctype`
- `iosstate` `_M_exception`
- `char_type` `_M_fill`
- `bool` `_M_fill_init`
- `fmtflags` `_M_flags`
- `streamsize` `_M_gcount`
- `locale` `_M_ios_locale`
- `_Words` `_M_local_word` [`_S_local_word_size`]
- `const __num_get_type` \* `_M_num_get`
- `const __num_put_type` \* `_M_num_put`
- `streamsize` `_M_precision`
- `basic_streambuf<_CharT, _Traits>` \* `_M_streambuf`
- `iosstate` `_M_streambuf_state`
- `basic_ostream<_CharT, _Traits>` \* `_M_tie`
- `streamsize` `_M_width`
- `_Words` \* `_M_word`
- `int` `_M_word_size`
- `_Words` `_M_word_zero`

### Friends

- class `sentry`
- class `sentry`

#### 5.392.1 Detailed Description

`template<typename _CharT, typename _Traits, typename _Alloc> class  
std::basic_stringstream<_CharT, _Traits, _Alloc>`

Controlling input and output for `std::string`.

This class supports reading from and writing to objects of type `std::basic_string`, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

Definition at line 478 of file `sstream`.

### 5.392.2 Member Typedef Documentation

**5.392.2.1** `template<typename _CharT, typename _Traits> typedef  
ctype<_CharT> std::basic_istream< _CharT, _Traits  
>::__ctype_type [inherited]`

These are non-standard types.

Reimplemented from `std::basic_ios< _CharT, _Traits >`.

Definition at line 73 of file `istream`.

**5.392.2.2** `template<typename _CharT, typename _Traits> typedef  
ctype<_CharT> std::basic_ostream< _CharT, _Traits  
>::__ctype_type [inherited]`

These are non-standard types.

Reimplemented from `std::basic_ios< _CharT, _Traits >`.

Definition at line 73 of file `ostream`.

**5.392.2.3** `template<typename _CharT, typename _Traits> typedef  
num_get<_CharT, istreambuf_iterator<_CharT, _Traits>  
> std::basic_istream< _CharT, _Traits >::__num_get_type  
[inherited]`

These are non-standard types.

Reimplemented from `std::basic_ios< _CharT, _Traits >`.

Definition at line 72 of file `istream`.

**5.392.2.4** `template<typename _CharT, typename _Traits> typedef  
num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>  
> std::basic_ostream< _CharT, _Traits >::__num_put_type  
[inherited]`

These are non-standard types.

Reimplemented in `std::basic_ostream< _CharT, _Traits >`, `std::basic_ostream< char, _Traits >`, and `std::basic_ostream< char >`.

Definition at line 86 of file `basic_ios.h`.

**5.392.2.5** `template<typename _CharT, typename _Traits> typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ostream<_CharT, _Traits>::__num_put_type [inherited]`

These are non-standard types.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Definition at line 72 of file `ostream`.

**5.392.2.6** `template<typename _CharT, typename _Traits> typedef _CharT std::basic_istream<_CharT, _Traits>::char_type [inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Reimplemented in `std::basic_ifstream<_CharT, _Traits>`, and `std::basic_istream<_CharT, _Traits, _Alloc>`.

Definition at line 61 of file `istream`.

**5.392.2.7** `template<typename _CharT, typename _Traits, typename _Alloc> typedef _CharT std::basic_stringstream<_CharT, _Traits, _Alloc>::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_iostream<_CharT, _Traits>`.

Definition at line 482 of file `sstream`.

**5.392.2.8** `typedef void(* std::ios_base::event_callback)(event, ios_base &, int) [inherited]`

The type of an event callback function.

#### Parameters

*event* One of the members of the event enum.

*ios\_base* Reference to the `ios_base` object.

*int* The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several [ios\\_base](#) and [basic\\_ios](#) functions, specifically [imbue\(\)](#), [copyfmt\(\)](#), and [~ios\(\)](#).

Definition at line 438 of file [ios\\_base.h](#).

#### **5.392.2.9 typedef \_Ios\_Fmtflags std::ios\_base::fmtflags [inherited]**

This is a bitmask type.

*\_Ios\_Fmtflags* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type *fmtflags* are:

- *boolalpha*
- *dec*
- *fixed*
- *hex*
- *internal*
- *left*
- *oct*
- *right*
- *scientific*
- *showbase*
- *showpoint*
- *showpos*
- *skipws*
- *unitbuf*
- *uppercase*
- *adjustfield*
- *basefield*
- *floatfield*

Definition at line 257 of file [ios\\_base.h](#).

**5.392.2.10** `template<typename _CharT, typename _Traits> typedef  
_Traits::int_type std::basic_istream< _CharT, _Traits >::int_type  
[inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_ios< \\_CharT, \\_Traits >](#).

Reimplemented in [std::basic\\_ifstream< \\_CharT, \\_Traits >](#), and [std::basic\\_istream< \\_CharT, \\_Traits, \\_Alloc >](#).

Definition at line 62 of file istream.

**5.392.2.11** `template<typename _CharT, typename _Traits, typename _Alloc  
> typedef traits_type::int_type std::basic_stringstream< _CharT,  
_Traits, _Alloc >::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_iostream< \\_CharT, \\_Traits >](#).

Definition at line 487 of file sstream.

**5.392.2.12** `typedef _Ios_Iostate std::ios_base::iostate [inherited]`

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 332 of file `ios_base.h`.



**5.392.2.13** `template<typename _CharT, typename _Traits, typename _Alloc  
> typedef traits_type::off_type std::basic_stringstream< _CharT,  
_Traits, _Alloc >::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_istream< \\_CharT, \\_Traits >](#).

Definition at line 489 of file sstream.

**5.392.2.14** `template<typename _CharT, typename _Traits> typedef  
_Traits::off_type std::basic_istream< _CharT, _Traits >::off_type  
[inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_ios< \\_CharT, \\_Traits >](#).

Reimplemented in [std::basic\\_ifstream< \\_CharT, \\_Traits >](#), and [std::basic\\_istream< \\_CharT, \\_Traits, \\_Alloc >](#).

Definition at line 64 of file istream.

**5.392.2.15** `typedef _Ios_Openmode std::ios_base::openmode [inherited]`

This is a bitmask type.

*\_Ios\_Openmode* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type *openmode* are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 363 of file ios\_base.h.

**5.392.2.16** `template<typename _CharT, typename _Traits, typename _Alloc> typedef traits_type::pos_type std::basic_stringstream<_CharT, _Traits, _Alloc>::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_istream<_CharT, _Traits>`.

Definition at line 488 of file `sstream`.

**5.392.2.17** `template<typename _CharT, typename _Traits> typedef _Traits::pos_type std::basic_istream<_CharT, _Traits>::pos_type [inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Reimplemented in `std::basic_ifstream<_CharT, _Traits>`, and `std::basic_istream<_CharT, _Traits, _Alloc>`.

Definition at line 63 of file `istream`.

**5.392.2.18** `typedef _Ios_Seekdir std::ios_base::seekdir [inherited]`

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 395 of file `ios_base.h`.

**5.392.2.19** `template<typename _CharT, typename _Traits> typedef _Traits std::basic_istream<_CharT, _Traits>::traits_type [inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which

are specific to the implementation.

Reimplemented from [std::basic\\_ios< \\_CharT, \\_Traits >](#).

Reimplemented in [std::basic\\_ifstream< \\_CharT, \\_Traits >](#), and [std::basic\\_istream< \\_CharT, \\_Traits, \\_Alloc >](#).

Definition at line 65 of file istream.

**5.392.2.20** `template<typename _CharT, typename _Traits, typename _Alloc  
> typedef _Traits std::basic_stringstream< _CharT, _Traits, _Alloc  
>::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic\\_iostream< \\_CharT, \\_Traits >](#).

Definition at line 483 of file sstream.

### 5.392.3 Member Enumeration Documentation

**5.392.3.1** `enum std::ios_base::event [inherited]`

The set of events that may be passed to an event callback.

erase\_event is used during ~ios() and copyfmt(). imbue\_event is used during [imbue\(\)](#). copyfmt\_event is used during copyfmt().

Definition at line 421 of file ios\_base.h.

### 5.392.4 Constructor & Destructor Documentation

**5.392.4.1** `template<typename _CharT, typename _Traits, typename  
_Alloc > std::basic_stringstream< _CharT, _Traits, _Alloc  
>::basic_stringstream ( ios_base::openmode __m =  
ios_base::out | ios_base::in ) [inline, explicit]`

Default constructor starts with an empty string buffer.

#### Parameters

*mode* Whether the buffer can read, or write, or both.

## 5.392 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference 2305

---

Initializes `sb` using `mode`, and passes `&sb` to the base class initializer. Does not allocate any buffer.

That's a lie. We initialize the base class with `NULL`, because the string class does its own memory management.

Definition at line 512 of file `sstream`.

References `std::basic_ios<_CharT, _Traits>::init()`.

**5.392.4.2** `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_stringstream<_CharT, _Traits, _Alloc>::basic_stringstream ( const __string_type & __str, ios_base::openmode __m = ios_base::out | ios_base::in ) [inline, explicit]`

Starts with an existing string buffer.

### Parameters

***str*** A string to copy as a starting buffer.

***mode*** Whether the buffer can read, or write, or both.

Initializes `sb` using `str` and `mode`, and passes `&sb` to the base class initializer.

That's a lie. We initialize the base class with `NULL`, because the string class does its own memory management.

Definition at line 528 of file `sstream`.

References `std::basic_ios<_CharT, _Traits>::init()`.

**5.392.4.3** `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_stringstream<_CharT, _Traits, _Alloc>::~~basic_stringstream ( ) [inline]`

The destructor does nothing.

The buffer is deallocated by the `stringbuf` object, not the formatting stream.

Definition at line 539 of file `sstream`.

### 5.392.5 Member Function Documentation

#### 5.392.5.1 `const locale& std::ios_base::_M_getloc ( ) const` [`inline`, `inherited`]

Locale access.

##### Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 708 of file `ios_base.h`.

Referenced by `std::money_get<_CharT, _InIter>::do_get()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_date()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_time()`, `std::time_get<_CharT, _InIter>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::time_put<_CharT, _OutIter>::do_put()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::time_put<_CharT, _OutIter>::put()`.

#### 5.392.5.2 `template<typename _CharT, typename _Traits> void` `std::basic_ostream<_CharT, _Traits>::_M_write ( const char_type` `* __s, streamsize __n )` [`inline`, `inherited`]

Simple insertion.

##### Parameters

*c* The character to insert.

##### Returns

`*this`

Tries to insert *c*.

##### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 289 of file `ostream`.

Referenced by `std::basic_ostream<_CharT, _Traits>::write()`.

**5.392.5.3** `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::bad ( ) const` [**inline, inherited**]

Fast error checking.

#### Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 203 of file `basic_ios.h`.

**5.392.5.4** `template<typename _CharT, typename _Traits> void std::basic_ios<_CharT, _Traits>::clear ( iostate __state = goodbit )` [**inherited**]

[Re]sets the error state.

#### Parameters

*state* The new state flag(s) to set.

See [std::ios\\_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

Definition at line 42 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::exceptions()`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::rdstate()`.

Referenced by `std::basic_ios<char, _Traits>::exceptions()`, `std::basic_fstream<_CharT, _Traits>::open()`, `std::basic_ofstream<_CharT, _Traits>::open()`, `std::basic_ifstream<_CharT, _Traits>::open()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ios<char, _Traits>::setstate()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

**5.392.5.5** `template<typename _CharT, typename _Traits> basic_ios<_CharT, _Traits> & std::basic_ios<_CharT, _Traits>::copyfmt ( const basic_ios<_CharT, _Traits> & __rhs )` [**inherited**]

Copies fields of `__rhs` into this.

#### Parameters

`__rhs` The source values for the copies.

#### Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy `exceptions()`.

Definition at line 64 of file `basic_ios.tcc`.

References `std::basic_ios< _CharT, _Traits >::exceptions()`, `std::basic_ios< _CharT, _Traits >::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios< _CharT, _Traits >::tie()`, and `std::ios_base::width()`.

#### 5.392.5.6 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::eof( ) const [inline, inherited]`

Fast error checking.

#### Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 182 of file `basic_ios.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

#### 5.392.5.7 `template<typename _CharT, typename _Traits> iostate std::basic_ios< _CharT, _Traits >::exceptions( ) const [inline, inherited]`

Throwing exceptions on errors.

### Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of [exceptions\(iostate\)](#) for the meaning of the return value.

Definition at line 214 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::clear()`, and `std::basic_ios<_CharT, _Traits>::copyfmt()`.

**5.392.5.8** `template<typename _CharT, typename _Traits> void std::basic_ios<_CharT, _Traits>::exceptions ( iostate __except ) [inline, inherited]`

Throwing exceptions on errors.

### Parameters

*except* The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type [std::ios\\_base::failure](#) is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
 std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

 std::ifstream f ("/etc/motd");

 std::cerr << "Setting badbit\n";
 f.setstate (std::ios_base::badbit);

 std::cerr << "Setting exception mask\n";
 f.exceptions (std::ios_base::badbit);
}
```

Definition at line 249 of file `basic_ios.h`.



**5.392.5.9** `template<typename _CharT, typename _Traits> bool std::basic_ios<  
_CharT, _Traits >::fail ( ) const [inline, inherited]`

Fast error checking.

**Returns**

True if either the badbit or the failbit is set.

Checking the badbit in [fail\(\)](#) is historical practice. Note that other iostate flags may also be set.

Definition at line 193 of file basic\_ios.h.

Referenced by std::basic\_ios< char, \_Traits >::operator void \*(), std::basic\_ios< char, \_Traits >::operator!(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_ostream< \_CharT, \_Traits >::tellp(), and std::regex\_traits< \_Ch\_type >::value().

**5.392.5.10** `template<typename _CharT, typename _Traits> char_type  
std::basic_ios< _CharT, _Traits >::fill ( ) const [inline,  
inherited]`

Retrieves the *empty* character.

**Returns**

The current fill character.

It defaults to a space ( ' ' ) in the current locale.

Definition at line 362 of file basic\_ios.h.

Referenced by std::basic\_ios< \_CharT, \_Traits >::copyfmt(), and std::basic\_ios< char, \_Traits >::fill().

**5.392.5.11** `template<typename _CharT, typename _Traits> char_type  
std::basic_ios< _CharT, _Traits >::fill ( char_type __ch )  
[inline, inherited]`

Sets a new *empty* character.

### Parameters

*ch* The new character.

### Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 382 of file `basic_ios.h`.

#### **5.392.5.12 fmtflags std::ios\_base::flags ( ) const [inline, inherited]**

Access to format flags.

### Returns

The format control flags for both input and output.

Definition at line 553 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, `std::operator>>()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

#### **5.392.5.13 fmtflags std::ios\_base::flags ( fmtflags \_\_fmtfl ) [inline, inherited]**

Setting new format flags all at once.

### Parameters

*fmtfl* The new flags to set.

### Returns

The previous format control flags.

This function overwrites all the format flags with *fmtfl*.

Definition at line 564 of file `ios_base.h`.

**5.392.5.14** `template<typename _CharT, typename _Traits > basic_ostream<  
_CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::flush  
( ) [inherited]`

Synchronizing the stream buffer.

**Returns**

\*this

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit.

Definition at line 213 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

Referenced by `std::flush()`.

**5.392.5.15** `template<typename _CharT, typename _Traits> streamsize  
std::basic_istream< _CharT, _Traits >::gcount( ) const  
[inline, inherited]`

Character counting.

**Returns**

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 251 of file istream.

**5.392.5.16** `template<typename _CharT, typename _Traits > basic_istream<  
_CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits  
>::get( void ) [inherited]`

Simple extraction.

**Returns**

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 238 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits >::\_M\_gcount, std::ios\_base::badbit, std::basic\_ios< \_CharT, \_Traits >::eof(), std::ios\_base::eofbit, std::ios\_base::failbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), and std::basic\_ios< \_CharT, \_Traits >::setstate().

**5.392.5.17** `template<typename _CharT, typename _Traits> basic_istream<  
_CharT, _Traits> & std::basic_istream< _CharT, _Traits>::get (  
char_type & __c ) [inherited]`

Simple extraction.

#### Parameters

*c* The character in which to store data.

#### Returns

\*this

Tries to extract a character and store it in *c*. If none are available, sets failbit and returns traits::eof().

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 274 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits >::\_M\_gcount, std::ios\_base::badbit, std::ios\_base::eofbit, std::ios\_base::failbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), and std::basic\_ios< \_CharT, \_Traits >::setstate().

**5.392.5.18** `template<typename _CharT, typename _Traits> __istream_type&  
std::basic_istream< _CharT, _Traits>::get ( char_type * __s,  
streamsize __n ) [inline, inherited]`

Simple multiple-character extraction.

#### Parameters

*s* Pointer to an array.

*n* Maximum number of characters to store in *s*.

**Returns**

\*this

Returns `get(s,n,widen('\n'))`.

Definition at line 335 of file istream.

**5.392.5.19** `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get ( __streambuf_type & __sb, char_type __delim ) [inherited]`

Extraction into another streambuf.

**Parameters**

*sb* A streambuf in which to store data.

*delim* A "stop" character.

**Returns**

\*this

Characters are extracted and inserted into *sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 358 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, `std::basic_streambuf< _CharT, _Traits >::snextc()`, and `std::basic_streambuf< _CharT, _Traits >::sputc()`.

**5.392.5.20** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::get ( __streambuf_type & __sb ) [inline, inherited]`

Extraction into another streambuf.

**Parameters**

*sb* A streambuf in which to store data.

**Returns**

\*this

Returns `get(sb, widen('\n'))`.

Definition at line 368 of file `istream`.

**5.392.5.21** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get ( char_type * __s, streamsize __n, char_type __delim ) [inherited]`

Simple multiple-character extraction.

**Parameters**

*s* Pointer to an array.

*n* Maximum number of characters to store in *s*.

*delim* A "stop" character.

**Returns**

\*this

Characters are extracted and stored into *s* until one of the following happens:

- *n*-1 characters are stored
- the input sequence reaches EOF
- the next character equals *delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 311 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

**5.392.5.22** `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits> & std::basic_istream< _CharT, _Traits>::getline( char_type * __s, streamsize __n, char_type __delim ) [inherited]`

String extraction.

#### Parameters

*s* A character array in which to store the data.

*n* Maximum number of characters to extract.

*delim* A "stop" character.

#### Returns

\*this

Extracts and stores characters into *s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals *delim*, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. *n*-1 characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 402 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits >::\_M\_gcount, std::ios\_base::badbit, std::basic\_ios< \_CharT, \_Traits >::eof(), std::ios\_base::eofbit, std::ios\_base::failbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), std::basic\_streambuf< \_CharT, \_Traits >::sbumpc(), std::basic\_ios< \_CharT, \_Traits >::setstate(), std::basic\_streambuf< \_CharT, \_Traits >::sgetc(), and std::basic\_streambuf< \_CharT, \_Traits >::snextc().

**5.392.5.23** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::getline ( char_type * __s, streamsize __n ) [inline, inherited]`

String extraction.

#### Parameters

*s* A character array in which to store the data.

*n* Maximum number of characters to extract.

#### Returns

\*this

Returns `getline(s,n,widen('\n'))`.

Definition at line 408 of file istream.

Referenced by std::basic\_istream< char >::getline().

**5.392.5.24** `locale std::ios_base::getloc ( ) const [inline, inherited]`

Locale access.

#### Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.



Definition at line 697 of file ios\_base.h.

Referenced by std::basic\_ios< \_CharT, \_Traits >::copyfmt(), std::money\_put< \_CharT, \_Outputter >::do\_put(), std::basic\_ios< \_CharT, \_Traits >::imbue(), std::operator>>(), and std::ws().

**5.392.5.25** `template<typename _CharT, typename _Traits> bool  
std::basic_ios< _CharT, _Traits >::good ( ) const [inline,  
inherited]`

Fast error checking.

#### Returns

True if no error flags are set.

A wrapper around rdstate.

Definition at line 172 of file basic\_ios.h.

Referenced by std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry(), and std::basic\_istream< \_CharT, \_Traits >::sentry::sentry().

**5.392.5.26** `template<typename _CharT, typename _Traits > basic_istream<  
_CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore  
( streamsize __n ) [inherited]`

Simple extraction.

#### Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 495 of file istream.tcc.

References std::basic\_istream< \_CharT, \_Traits >::\_M\_gcount, std::ios\_base::badbit, std::basic\_ios< \_CharT, \_Traits >::eof(), std::ios\_base::eofbit, std::ios\_base::goodbit, std::basic\_ios< \_CharT, \_Traits >::rdbuf(), std::basic\_ios< \_CharT, \_Traits >::setstate(), std::basic\_streambuf< \_CharT, \_Traits >::sgetc(), and std::basic\_streambuf< \_CharT, \_Traits >::snextc().

**5.392.5.27** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore( void ) [inherited]`

Discarding characters.

#### Parameters

*n* Number of characters to discard.

*delim* A "stop" character.

#### Returns

\*this

Extracts characters and throws them away until one of the following happens:

- if *n* != `std::numeric_limits<int>::max()`, *n* characters are extracted
- the input sequence reaches end-of-file
- the next character equals *delim* (in this case, the character is extracted); note that this condition will never occur if *delim* equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 462 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sbumpc()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.392.5.28** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore( streamsize __n, int_type __delim ) [inherited]`

Simple extraction.

#### Returns

A character, or `eof()`.

### 5.392 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference 2320

---

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 557 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sbumpc()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

#### 5.392.5.29 `template<typename _CharT, typename _Traits> locale std::basic_ios<_CharT, _Traits>::imbue ( const locale & __loc ) [inherited]`

Moves to a new locale.

##### Parameters

*loc* The new locale.

##### Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Reimplemented from `std::ios_base`.

Definition at line 115 of file `basic_ios.tcc`.

References `std::ios_base::getloc()`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

Referenced by `std::operator<<()`.

#### 5.392.5.30 `template<typename _CharT, typename _Traits> void std::basic_ios<_CharT, _Traits>::init ( basic_streambuf<_CharT, _Traits> * __sb ) [protected, inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 127 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

Referenced by std::basic\_fstream< \_CharT, \_Traits >::basic\_fstream(), std::basic\_ifstream< \_CharT, \_Traits >::basic\_ifstream(), std::basic\_istream< char, \_Traits >::basic\_istream(), std::basic\_istream< char >::basic\_istream(), std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >::basic\_istreamstream(), std::basic\_ofstream< \_CharT, \_Traits >::basic\_ofstream(), std::basic\_ostream< char >::basic\_ostream(), std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >::basic\_ostreamstream(), and std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >::basic\_stringstream().

**5.392.5.31 long& std::ios\_base::iword ( int \_\_ix ) [inline, inherited]**

Access to integer array.

**Parameters**

***\_\_ix*** Index into the array.

**Returns**

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 743 of file ios\_base.h.

**5.392.5.32 template<typename \_CharT, typename \_Traits> char  
std::basic\_istream< \_CharT, \_Traits >::narrow ( char\_type \_\_c, char  
\_\_dfault ) const [inline, inherited]**

Squeezes characters.

**Parameters**

***c*** The character to narrow.

***dfault*** The character to narrow.

**Returns**

The narrowed character.

### 5.392 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference 2322

---

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, default)
```

Additional notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 422 of file `basic_ios.h`.

**5.392.5.33** `template<typename _CharT, typename _Traits> std::basic_ios<_CharT, _Traits>::operator void * ( ) const` [`inline`, `inherited`]

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 113 of file `basic_ios.h`.

**5.392.5.34** `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::operator! ( ) const` [`inline`, `inherited`]

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 117 of file `basic_ios.h`.

**5.392.5.35** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< ( float __f )` [`inline`, `inherited`]

Basic arithmetic inserters.

#### Parameters

`A` variable of builtin type.

#### Returns

`*this` if successful

## 5.392 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference 2323

---

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 215 of file `ostream`.

**5.392.5.36** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< ( bool __n ) [inline, inherited]`

Basic arithmetic inserters.

### Parameters

*A* variable of builtin type.

### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 175 of file `ostream`.

**5.392.5.37** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< ( unsigned short __n ) [inline, inherited]`

Basic arithmetic inserters.

### Parameters

*A* variable of builtin type.

### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 182 of file `ostream`.

**5.392.5.38** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< ( __ostream_type &(*)(__ostream_type &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the `iomanip` header.

Definition at line 110 of file `ostream`.

**5.392.5.39** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< ( __ios_type &(*)(__ios_type &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the `iomanip` header.

Definition at line 119 of file `ostream`.

**5.392.5.40** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< ( ios_base &(*)(ios_base &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the `iomanip` header.

Definition at line 129 of file `ostream`.

**5.392.5.41** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< ( unsigned long __n ) [inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 171 of file ostream.

**5.392.5.42** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<<( short __n ) [inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 94 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

**5.392.5.43** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<<( int __n ) [inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.



### Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 108 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

**5.392.5.44** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( long long __n ) [inline, inherited]`

Basic arithmetic inserters.

### Parameters

*A* variable of builtin type.

### Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 202 of file ostream.

**5.392.5.45** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( const void * __p ) [inline, inherited]`

Basic arithmetic inserters.

### Parameters

*A* variable of builtin type.

### Returns

\*this if successful

**5.392 std::basic\_stringstream<\_CharT, \_Traits, \_Alloc> Class Template Reference** 2327

---

These functions use the stream's current locale (specifically, the [num\\_get](#) facet) to perform numeric formatting.

Definition at line 227 of file ostream.

**5.392.5.46** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream<_CharT, _Traits>::operator<<( double __f  
) [inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the [num\\_get](#) facet) to perform numeric formatting.

Definition at line 211 of file ostream.

**5.392.5.47** `template<typename _CharT, typename _Traits> __ostream_type&  
std::basic_ostream<_CharT, _Traits>::operator<<( long double  
__f ) [inline, inherited]`

Basic arithmetic inserters.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the [num\\_get](#) facet) to perform numeric formatting.

Definition at line 223 of file ostream.

**5.392.5.48** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<<( __streambuf_type * __sb ) [inherited]`

Extracting from another streambuf.

#### Parameters

*sb* A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from *sb* and inserted into *\*this* until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from *sb*, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 122 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.392.5.49** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<( long __n ) [inline, inherited]`

Basic arithmetic inserters.

#### Parameters

*A* variable of builtin type.

#### Returns

*\*this* if successful

## 5.392 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference 2329

---

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 167 of file `ostream`.

**5.392.5.50** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< ( unsigned int __n ) [inline, inherited]`

Basic arithmetic inserters.

### Parameters

*A* variable of builtin type.

### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 193 of file `ostream`.

**5.392.5.51** `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< ( unsigned long long __n ) [inline, inherited]`

Basic arithmetic inserters.

### Parameters

*A* variable of builtin type.

### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 206 of file `ostream`.

**5.392.5.52** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::operator>> ( short & __n ) [inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 116 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get<_CharT, _InIter>::get()`, `std::ios_base::goodbit`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.392.5.53** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( bool & __n ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 169 of file `istream`.

**5.392.5.54** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( float & __f ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 205 of file istream.

**5.392.5.55** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( ios_base &(*)(ios_base &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iomanip` header.

Definition at line 133 of file istream.

**5.392.5.56** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( double &__f ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 209 of file istream.

**5.392.5.57** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( long double & __f ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

`A` variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 213 of file `istream`.

**5.392.5.58** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( unsigned short & __n ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

`A` variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 176 of file `istream`.

**5.392.5.59** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( long & __n ) [inline, inherited]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 187 of file istream.

**5.392.5.60** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( void *& __p ) [inline, inherited]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 217 of file istream.

**5.392.5.61** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( long long & __n ) [inline, inherited]`

Basic arithmetic extractors.

**Parameters**

*A* variable of builtin type.

**Returns**

\*this if successful



These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 196 of file `istream`.

**5.392.5.62** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::operator>>(int & __n) [inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 161 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get<_CharT, _InIter>::get()`, `std::ios_base::goodbit`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.392.5.63** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::operator>>(_streambuf_type * __sb) [inherited]`

Extracting into another streambuf.

#### Parameters

*sb* A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the *sb* streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 206 of file istream.tcc.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.392.5.64** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( unsigned int & __n ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 183 of file istream.

**5.392.5.65** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( __istream_type &(*)(__istream_type &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 122 of file istream.

**5.392.5.66** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( __ios_type &(*)(__ios_type &) __pf ) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 126 of file `istream`.

**5.392.5.67** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( unsigned long & __n ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 191 of file `istream`.

**5.392.5.68** `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> ( unsigned long long & __n ) [inline, inherited]`

Basic arithmetic extractors.

#### Parameters

*A* variable of builtin type.

#### Returns

`*this` if successful

### 5.392 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference 2337

---

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 200 of file `istream`.

**5.392.5.69** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits>::peek( void ) [inherited]`

Looking ahead in the stream.

#### Returns

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 622 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.392.5.70** `streamsize std::ios_base::precision( ) const [inline, inherited]`

Flags access.

#### Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 623 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, and `std::operator<<()`.

**5.392.5.71** `streamsize std::ios_base::precision( streamsize __prec ) [inline, inherited]`

Changing flags.

#### Parameters

*prec* The new precision value.

#### Returns

The previous value of `precision()`.

Definition at line 632 of file `ios_base.h`.

**5.392.5.72** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::put (char_type __c) [inherited]`

Simple insertion.

#### Parameters

*c* The character to insert.

#### Returns

`*this`

Tries to insert *c*.

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 151 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

Referenced by `std::endl()`, and `std::ends()`.

**5.392.5.73** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::putback (char_type __c) [inherited]`

Unextracting a single character.

**Parameters**

*c* The character to push back into the input stream.

**Returns**

\*this

If `rdbuf()` is not null, calls `rdbuf() -> sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

**Note**

Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 713 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::rdstate()`, `std::basic_ios< _CharT, _Traits >::setstate()`, and `std::basic_streambuf< _CharT, _Traits >::sputbackc()`.

Referenced by `std::operator>>()`.

**5.392.5.74 void\*& std::ios\_base::pword ( int \_\_ix ) [inline, inherited]**

Access to void pointer array.

**Parameters**

*\_\_ix* Index into the array.

**Returns**

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 764 of file `ios_base.h`.

**5.392.5.75** `template<typename _CharT, typename _Traits, typename _Alloc  
> __stringbuf_type* std::basic_stringstream<_CharT, _Traits,  
_Alloc>::rdbuf( ) const [inline]`

Accessing the underlying buffer.

#### Returns

The current `basic_stringbuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Definition at line 550 of file `sstream`.

**5.392.5.76** `template<typename _CharT, typename _Traits> basic_streambuf<  
_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rdbuf (  
basic_streambuf<_CharT, _Traits> * __sb ) [inherited]`

Changing the underlying buffer.

#### Parameters

*sb* The new stream buffer.

#### Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream foo; // or some other derived type
std::streambuf* p =;

foo.ios::rdbuf(p); // ios == basic_ios<char>
```

Definition at line 54 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::clear()`.

**5.392.5.77** `template<typename _CharT, typename _Traits> iostate  
std::basic_ios<_CharT, _Traits>::rdstate ( ) const [inline,  
inherited]`

Returns the error state of the stream buffer.

#### Returns

A bit pattern (well, isn't everything?)

See [std::ios\\_base::iostate](#) for the possible bit values. Most users will call one of the interpreting wrappers, e.g., [good\(\)](#).

Definition at line 129 of file `basic_ios.h`.

Referenced by `std::basic_ios<char, _Traits>::bad()`, `std::basic_ios<_CharT, _Traits>::clear()`, `std::basic_ios<char, _Traits>::eof()`, `std::basic_ios<char, _Traits>::fail()`, `std::basic_ios<char, _Traits>::good()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ios<char, _Traits>::setstate()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

**5.392.5.78** `template<typename _CharT, typename _Traits> basic_istream<  
_CharT, _Traits> &std::basic_istream<_CharT, _Traits>::read (  
char_type * __s, streamsize __n ) [inherited]`

Extraction without delimiters.

#### Parameters

*s* A character array.

*n* Maximum number of characters to store.

#### Returns

\*this

If the stream state is [good\(\)](#), extracts characters and stores them into *s* until one of the following happens:

- *n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.



### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 652 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.392.5.79** `template<typename _CharT, typename _Traits > streamsize  
std::basic_istream< _CharT, _Traits >::readsome ( char_type *  
__s, streamsize __n ) [inherited]`

Extraction until the buffer is exhausted, but no more.

### Parameters

- s* A character array.
- n* Maximum number of characters to store.

### Returns

The number of characters extracted.

Extracts characters and stores them into *s* depending on the number of characters remaining in the streambuf's buffer, `rdbuf() -> in_avail()`, called *A* here:

- if *A* == -1, sets eofbit and extracts no characters
- if *A* == 0, extracts no characters
- if *A* > 0, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 681 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::min()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.392.5.80** `void std::ios_base::register_callback ( event_callback __fn, int  
__index ) [inherited]`

Add the callback `__fn` with parameter `__index`.

#### Parameters

`__fn` The function to add.

`__index` The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**5.392.5.81** `template<typename _CharT, typename _Traits > basic_istream<  
_CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg  
( off_type __off, ios_base::seekdir __dir ) [inherited]`

Changing the current read position.

#### Parameters

`off` A file offset object.

`dir` The direction in which to seek.

#### Returns

`*this`

If `fail()` is not true, calls `rddbuf()`  $\rightarrow$  `pubseekoff(off, dir)`. If that function fails, sets failbit.

#### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 886 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::ios_base::eofbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios< _CharT, _Traits >::rddbuf()`, `std::basic_ios< _CharT, _Traits >::rdstate()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**5.392.5.82** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg ( pos_type __pos ) [inherited]`

Changing the current read position.

#### Parameters

*pos* A file position object.

#### Returns

\*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.

#### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 847 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::clear()`, `std::ios_base::eofbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::rdstate()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.392.5.83** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp ( off_type __off, ios_base::seekdir __dir ) [inherited]`

Changing the current write position.

#### Parameters

*off* A file offset object.

*dir* The direction in which to seek.

#### Returns

\*this

If `fail()` is not true, calls `rdbuf() ->pubseekoff(off, dir)`. If that function fails, sets failbit.

Definition at line 292 of file ostream.tcc.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.392.5.84** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp( pos_type __pos ) [inherited]`

Changing the current write position.

#### Parameters

*pos* A file position object.

#### Returns

\*this

If `fail()` is not true, calls `rdbuf() ->pubseekpos(pos)`. If that function fails, sets failbit.

Definition at line 260 of file ostream.tcc.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.392.5.85** `fmtflags std::ios_base::setf( fmtflags __fmtfl ) [inline, inherited]`

Setting new format flags.

#### Parameters

*fmtfl* Additional flags to set.

#### Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 580 of file ios\_base.h.

Referenced by `std::dec()`, `std::fixed()`, `std::hex()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

**5.392.5.86** `fmtflags std::ios_base::setf ( fmtflags __fmtfl, fmtflags __mask )`  
`[inline, inherited]`

Setting new format flags.

#### Parameters

*fmtfl* Additional flags to set.

*mask* The flags mask for *fmtfl*.

#### Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 597 of file ios\_base.h.

**5.392.5.87** `template<typename _CharT, typename _Traits> void`  
`std::basic_ios< _CharT, _Traits >::setstate ( iostate __state )`  
`[inline, inherited]`

Sets additional flags in the error state.

#### Parameters

*state* The additional state flag(s) to set.

See `std::ios_base::iostate` for the possible bit values.

Definition at line 149 of file basic\_ios.h.

Referenced by `std::basic_ostream< char >::_M_write()`, `std::basic_fstream< _CharT, _Traits >::close()`, `std::basic_ofstream< _CharT, _Traits >::close()`, `std::basic_ifstream< _CharT, _Traits >::close()`, `std::basic_ostream< _CharT, _Traits >::flush()`,

std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_fstream< \_CharT, \_Traits >::open(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::sync(), std::basic\_istream< \_CharT, \_Traits >::unget(), and std::ws().

**5.392.5.88** `template<typename _CharT, typename _Traits, typename _Alloc  
> __string_type std::basic_stringstream< _CharT, _Traits, _Alloc  
>::str ( ) const [inline]`

Copying out the string buffer.

#### Returns

`rddbuf () -> str ()`

Definition at line 558 of file sstream.

**5.392.5.89** `template<typename _CharT, typename _Traits, typename _Alloc  
> void std::basic_stringstream< _CharT, _Traits, _Alloc >::str (   
const __string_type & __s ) [inline]`

Setting a new buffer.

#### Parameters

`s` The string to use as a new sequence.

Calls `rddbuf () -> str (s)`.

Definition at line 568 of file sstream.

**5.392.5.90** `template<typename _CharT, typename _Traits> int  
std::basic_istream<_CharT, _Traits>::sync ( void )  
[inherited]`

Synchronizing the stream buffer.

#### Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

#### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 783 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_streambuf<_CharT, _Traits>::pubsync()`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**5.392.5.91** `static bool std::ios_base::sync_with_stdio ( bool __sync = true )  
[static, inherited]`

Interaction with the standard C I/O objects.

#### Parameters

*sync* Whether to synchronize or not.

#### Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

**5.392.5.92** `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::pos_type std::basic_istream<_CharT, _Traits>::tellg( void ) [inherited]`

Getting the current read position.

#### Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, in)`.

#### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 819 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::in`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

**5.392.5.93** `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>::pos_type std::basic_ostream<_CharT, _Traits>::tellp( ) [inherited]`

Getting the current write position.

#### Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, out)`.

Definition at line 239 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::out`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.



**5.392.5.94** `template<typename _CharT, typename _Traits>  
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits  
>::tie( basic_ostream<_CharT, _Traits> * __tiestr ) [inline,  
inherited]`

Ties this stream to an output stream.

#### Parameters

*tiestr* The output stream.

#### Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see [tie\(\)](#) for more.

Definition at line 299 of file `basic_ios.h`.

**5.392.5.95** `template<typename _CharT, typename _Traits>  
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits  
>::tie( ) const [inline, inherited]`

Fetches the current *tied* stream.

#### Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 287 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

**5.392.5.96** `template<typename _CharT, typename _Traits> basic_istream<  
_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::unget  
( void ) [inherited]`

Unextracting the previous character.

**Returns**

\*this

If `rdbuf()` is not null, calls `rdbuf() -> sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

**Note**

Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 748 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::clear()`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::rdstate()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sungetc()`.

**5.392.5.97 void std::ios\_base::unsetf ( fmtflags \_\_mask ) [inline, inherited]**

Clearing format flags.

**Parameters**

*mask* The flags to unset.

This function clears *mask* in the format flags.

Definition at line 612 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

**5.392.5.98 template<typename \_CharT, typename \_Traits> char\_type  
std::basic\_ios<\_CharT, \_Traits>::widen ( char \_\_c ) const  
[inline, inherited]**

Widens characters.

**Parameters**

*c* The character to widen.

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 441 of file `basic_ios.h`.

Referenced by `std::endl()`, `std::basic_ios<char, _Traits>::fill()`, `std::basic_istream<char>::get()`, `std::basic_istream<char>::getline()`, `std::getline()`, and `std::operator>>()`.

**5.392.5.99 streamsize std::ios\_base::width ( streamsize \_\_wide ) [inline, inherited]**

Changing flags.

**Parameters**

*wide* The new width value.

**Returns**

The previous value of `width()`.

Definition at line 655 of file `ios_base.h`.

**5.392.5.100 streamsize std::ios\_base::width ( ) const [inline, inherited]**

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 646 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_put<_CharT, _Outputter>::do_put()`, and `std::operator>>()`.

**5.392.5.101** `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::write ( const char_type * __s, streamsize __n )`  
[*inherited*]

Character string insertion.

#### Parameters

- s* The array to insert.
- n* Maximum number of characters to insert.

#### Returns

\*this

Characters are copied from *s* and inserted into the stream until one of the following happens:

- *n* characters are inserted
- inserting into the output sequence fails (in this case, `badbit` will be set in the stream's error state)

#### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 185 of file `ostream.tcc`.

References `std::basic_ostream< _CharT, _Traits >::_M_write()`, and `std::ios_base::badbit`.

**5.392.5.102** `static int std::ios_base::xalloc ( ) throw ()` [*static*, *inherited*]

Access to unique indices.

#### Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

### 5.392.6 Member Data Documentation

#### 5.392.6.1 `template<typename _CharT, typename _Traits> streamsize std::basic_istream< _CharT, _Traits >::_M_gcount` `[protected, inherited]`

The number of characters extracted in the previous unformatted function; see [gcount\(\)](#).

Definition at line 81 of file `istream`.

Referenced by `std::basic_istream< char >::gcount()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::basic_istream< char >::~~basic_istream()`.

#### 5.392.6.2 `const fmtflags std::ios_base::adjustfield` `[static, inherited]`

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Definition at line 312 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

#### 5.392.6.3 `const openmode std::ios_base::app` `[static, inherited]`

Seek to end before each write.

Definition at line 366 of file `ios_base.h`.

**5.392.6.4 const openmode std::ios\_base::ate [static, inherited]**

Open and seek to end immediately after opening.

Definition at line 369 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::open().

**5.392.6.5 const iostate std::ios\_base::badbit [static, inherited]**

Indicates a loss of integrity in an input or output sequence (such /// as an irrecoverable read error from a file).

Definition at line 336 of file ios\_base.h.

Referenced by std::basic\_ostream< char >::\_M\_write(), std::basic\_ios< char, \_Traits >::bad(), std::basic\_ios< \_CharT, \_Traits >::clear(), std::basic\_ios< char, \_Traits >::fail(), std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_ios< \_CharT, \_Traits >::init(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::operator<<(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::sync(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_ostream< \_CharT, \_Traits >::tellp(), std::basic\_istream< \_CharT, \_Traits >::unget(), std::basic\_ostream< \_CharT, \_Traits >::write(), and std::basic\_ostream< \_CharT, \_Traits >::sentry::~sentry().

**5.392.6.6 const fmtflags std::ios\_base::basefield [static, inherited]**

A mask of dec|oct|hex. Useful for the 2-arg form of setf.

Definition at line 315 of file ios\_base.h.

Referenced by std::dec(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::hex(), std::oct(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

**5.392.6.7 const seekdir std::ios\_base::beg [static, inherited]**

Request a seek relative to the beginning of the stream.

Definition at line 398 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::seekpos().

**5.392.6.8 const openmode std::ios\_base::binary [static, inherited]**

Perform input and output in binary mode (as opposed to text mode). /// This is probably not what you think it is; see /// <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 374 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::showmanyc().

**5.392.6.9 const fmtflags std::ios\_base::boolalpha [static, inherited]**

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 260 of file ios\_base.h.

Referenced by std::boolalpha(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::noboolalpha().

**5.392.6.10 const seekdir std::ios\_base::cur [static, inherited]**

Request a seek relative to the current position within the sequence.

Definition at line 401 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::imbue(), std::basic\_filebuf< \_CharT, \_Traits >::overflow(), std::basic\_filebuf< \_CharT, \_Traits >::pbackfail(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_filebuf< \_CharT, \_Traits >::seekoff(), std::basic\_istream< \_CharT, \_Traits >::tellg(), and std::basic\_ostream< \_CharT, \_Traits >::tellp().

**5.392.6.11 const fmtflags std::ios\_base::dec [static, inherited]**

Converts integer input or generates integer output in decimal base.

Definition at line 263 of file ios\_base.h.

Referenced by std::dec().

**5.392.6.12 const seekdir std::ios\_base::end [static, inherited]**

Request a seek relative to the current end of the sequence.

Definition at line 404 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::open(), and std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff().

**5.392.6.13 const iostate std::ios\_base::eofbit [static, inherited]**

Indicates that an input operation reached the end of an input sequence.

Definition at line 339 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_date(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::eof(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::unget(), and std::ws().

**5.392.6.14 const iostate std::ios\_base::failbit [static, inherited]**

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.



Definition at line 344 of file ios\_base.h.

Referenced by std::basic\_fstream< \_CharT, \_Traits >::close(), std::basic\_ofstream< \_CharT, \_Traits >::close(), std::basic\_ifstream< \_CharT, \_Traits >::close(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::fail(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_fstream< \_CharT, \_Traits >::open(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry(), and std::basic\_istream< \_CharT, \_Traits >::sentry::sentry().

**5.392.6.15 const fmtflags std::ios\_base::fixed [static, inherited]**

Generate floating-point output in fixed-point notation.

Definition at line 266 of file ios\_base.h.

Referenced by std::fixed().

**5.392.6.16 const fmtflags std::ios\_base::floatfield [static, inherited]**

A mask of scientific|fixed. Useful for the 2-arg form of setf.

Definition at line 318 of file ios\_base.h.

Referenced by std::fixed(), and std::scientific().

**5.392.6.17 const iostate std::ios\_base::goodbit [static, inherited]**

Indicates all is well.

Definition at line 347 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(),

std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_ios< \_CharT, \_Traits >::init(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::sync(), and std::basic\_istream< \_CharT, \_Traits >::unget().

**5.392.6.18 const fmtflags std::ios\_base::hex [static, inherited]**

Converts integer input or generates integer output in hexadecimal base.

Definition at line 269 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), std::hex(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

**5.392.6.19 const openmode std::ios\_base::in [static, inherited]**

Open for input. Default for ifstream and fstream.

Definition at line 377 of file ios\_base.h.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::\_M\_set\_buffer(), std::basic\_ifstream< \_CharT, \_Traits >::open(), std::basic\_filebuf< \_CharT, \_Traits >::pbackfail(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::showmanyc(), std::basic\_filebuf< \_CharT, \_Traits >::showmanyc(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::underflow(), std::basic\_filebuf< \_CharT, \_Traits >::underflow(), and std::basic\_filebuf< \_CharT, \_Traits >::xsgetn().

**5.392.6.20 const fmtflags std::ios\_base::internal [static, inherited]**

Adds fill characters at a designated internal point in certain /// generated output, or identical to right if no such point is /// designated.

## 5.392 `std::basic_stringstream< _CharT, _Traits, _Alloc >` Class Template Reference 2360

---

Definition at line 274 of file `ios_base.h`.

Referenced by `std::internal()`.

### 5.392.6.21 `const fmtflags std::ios_base::left` `[static, inherited]`

Adds fill characters on the right (final positions) of certain `///` generated output. (I.e., the thing you print is flush left.).

Definition at line 278 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, and `std::left()`.

### 5.392.6.22 `const fmtflags std::ios_base::oct` `[static, inherited]`

Converts integer input or generates integer output in octal base.

Definition at line 281 of file `ios_base.h`.

Referenced by `std::oct()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

### 5.392.6.23 `const openmode std::ios_base::out` `[static, inherited]`

Open for output. Default for `ofstream` and `fstream`.

Definition at line 380 of file `ios_base.h`.

Referenced by `std::basic_filebuf< char_type, traits_type >::M_set_buffer()`, `std::basic_ofstream< _CharT, _Traits >::open()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

### 5.392.6.24 `const fmtflags std::ios_base::right` `[static, inherited]`

Adds fill characters on the left (initial positions) of certain `///` generated output. (I.e., the thing you print is flush right.).

Definition at line 285 of file `ios_base.h`.

Referenced by std::right().

**5.392.6.25 const fmtflags std::ios\_base::scientific [static, inherited]**

Generates floating-point output in scientific notation.

Definition at line 288 of file ios\_base.h.

Referenced by std::scientific().

**5.392.6.26 const fmtflags std::ios\_base::showbase [static, inherited]**

Generates a prefix indicating the numeric base of generated integer /// output.

Definition at line 292 of file ios\_base.h.

Referenced by std::noshowbase(), and std::showbase().

**5.392.6.27 const fmtflags std::ios\_base::showpoint [static, inherited]**

Generates a decimal-point character unconditionally in generated /// floating-point output.

Definition at line 296 of file ios\_base.h.

Referenced by std::noshowpoint(), and std::showpoint().

**5.392.6.28 const fmtflags std::ios\_base::showpos [static, inherited]**

Generates a + sign in non-negative generated numeric output.

Definition at line 299 of file ios\_base.h.

Referenced by std::noshowpos(), and std::showpos().

**5.392.6.29 const fmtflags std::ios\_base::skipws [static, inherited]**

Skips leading white space before certain input operations.

Definition at line 302 of file `ios_base.h`.

Referenced by `std::noskipws()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, and `std::skipws()`.

#### 5.392.6.30 `const openmode std::ios_base::trunc` [`static`, `inherited`]

Open for input. Default for `ofstream`.

Definition at line 383 of file `ios_base.h`.

#### 5.392.6.31 `const fmtflags std::ios_base::unitbuf` [`static`, `inherited`]

Flushes output after each output operation.

Definition at line 305 of file `ios_base.h`.

Referenced by `std::nunitbuf()`, `std::unitbuf()`, and `std::basic_ostream< _CharT, _Traits >::sentry::~sentry()`.

#### 5.392.6.32 `const fmtflags std::ios_base::uppercase` [`static`, `inherited`]

Replaces certain lowercase letters with their uppercase equivalents /// in generated output.

Definition at line 309 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, `std::nouppercase()`, and `std::uppercase()`.

The documentation for this class was generated from the following file:

- [sstream](#)

### 5.393 `std::bernoulli_distribution` Class Reference

A Bernoulli random number distribution.

#### Classes

- struct [param\\_type](#)

## Public Types

- typedef bool [result\\_type](#)

## Public Member Functions

- [bernoulli\\_distribution](#) (double \_\_p=0.5)
- **bernoulli\_distribution** (const [param\\_type](#) &\_\_p)
- [result\\_type](#) max () const
- [result\\_type](#) min () const
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) operator() (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator**() (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- double [p](#) () const
- void [param](#) (const [param\\_type](#) &\_\_param)
- [param\\_type](#) [param](#) () const
- void [reset](#) ()

### 5.393.1 Detailed Description

A Bernoulli random number distribution. Generates a sequence of true and false values with likelihood  $p$  that true will come up and  $(1 - p)$  that false will appear.

Definition at line 3217 of file random.h.

### 5.393.2 Member Typedef Documentation

#### 5.393.2.1 typedef bool std::bernoulli\_distribution::result\_type

The type of the range of the distribution.

Definition at line 3221 of file random.h.

### 5.393.3 Constructor & Destructor Documentation

#### 5.393.3.1 std::bernoulli\_distribution::bernoulli\_distribution ( double \_\_p = 0.5 ) [inline, explicit]

Constructs a Bernoulli distribution with likelihood  $p$ .

### Parameters

**\_\_p** [IN] The likelihood of a true result being returned. Must be in the interval[0, 1].

Definition at line 3254 of file random.h.

### 5.393.4 Member Function Documentation

#### 5.393.4.1 result\_type std::bernoulli\_distribution::max ( ) const [inline]

Returns the least upper bound value of the distribution.

Definition at line 3304 of file random.h.

#### 5.393.4.2 result\_type std::bernoulli\_distribution::min ( ) const [inline]

Returns the greatest lower bound value of the distribution.

Definition at line 3297 of file random.h.

#### 5.393.4.3 template<typename \_UniformRandomNumberGenerator > result\_type std::bernoulli\_distribution::operator() ( \_UniformRandomNumberGenerator & \_\_urng ) [inline]

Generating functions.

Definition at line 3312 of file random.h.

References operator()(), and param().

Referenced by operator()().

#### 5.393.4.4 double std::bernoulli\_distribution::p ( ) const [inline]

Returns the p parameter of the distribution.

Definition at line 3275 of file random.h.

**5.393.4.5** `void std::bernoulli_distribution::param ( const param_type & __param ) [inline]`

Sets the parameter set of the distribution.

#### Parameters

`__param` The new parameter set of the distribution.

Definition at line 3290 of file `random.h`.

**5.393.4.6** `param_type std::bernoulli_distribution::param ( ) const [inline]`

Returns the parameter set of the distribution.

Definition at line 3282 of file `random.h`.

Referenced by `operator()()`, `std::operator==()`, and `std::operator>>()`.

**5.393.4.7** `void std::bernoulli_distribution::reset ( ) [inline]`

Resets the distribution state.

Does nothing for a Bernoulli distribution.

Definition at line 3269 of file `random.h`.

The documentation for this class was generated from the following file:

- [random.h](#)

## 5.394 `std::bernoulli_distribution::param_type` Struct Reference

### Public Types

- typedef [bernoulli\\_distribution](#) `distribution_type`

### Public Member Functions

- `param_type` (double `__p`=0.5)
- double `p` () const



### Friends

- `bool operator==(const param\_type &__p1, const param\_type &__p2)`

#### 5.394.1 Detailed Description

Parameter type.

Definition at line 3223 of file random.h.

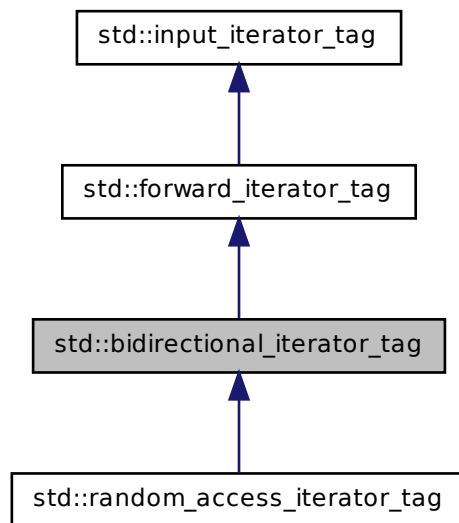
The documentation for this struct was generated from the following file:

- [random.h](#)

### 5.395 std::bidirectional\_iterator\_tag Struct Reference

Bidirectional iterators support a superset of forward iterator /// operations.

Inheritance diagram for std::bidirectional\_iterator\_tag:

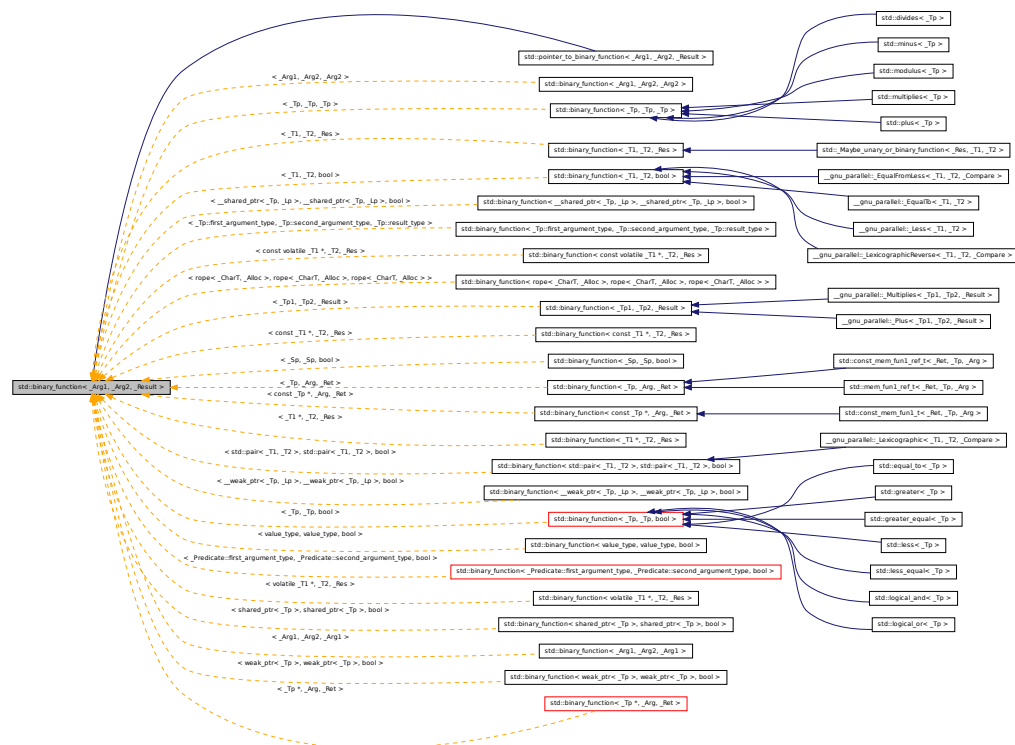


## 2367

Bidirectional iterators support a superset of forward iterator `///` operations.  
Definition at line 100 of file `stl_iterator_base_types.h`.  
The documentation for this struct was generated from the following file:

- `stl_iterator_base_types.h`

Inheritance diagram for `std::binary_function<_Arg1, _Arg2, _Result>`:



- typedef \_Arg1 first\_argument\_type

- typedef \_Result [result\\_type](#)
- typedef \_Arg2 [second\\_argument\\_type](#)

#### **5.396.1 Detailed Description**

**template<typename \_Arg1, typename \_Arg2, typename \_Result> struct std::binary\_function< \_Arg1, \_Arg2, \_Result >**

This is one of the [functor base classes](#).

Definition at line 115 of file stl\_function.h.

#### **5.396.2 Member Typedef Documentation**

**5.396.2.1 template<typename \_Arg1, typename \_Arg2, typename \_Result>  
typedef \_Arg1 std::binary\_function< \_Arg1, \_Arg2, \_Result  
>::first\_argument\_type**

`first_argument_type` is the type of the first argument

Definition at line 118 of file stl\_function.h.

**5.396.2.2 template<typename \_Arg1, typename \_Arg2, typename \_Result>  
typedef \_Result std::binary\_function< \_Arg1, \_Arg2, \_Result  
>::result\_type**

`result_type` is the return type

Definition at line 124 of file stl\_function.h.

**5.396.2.3 template<typename \_Arg1, typename \_Arg2, typename \_Result>  
typedef \_Arg2 std::binary\_function< \_Arg1, \_Arg2, \_Result  
>::second\_argument\_type**

`second_argument_type` is the type of the second argument

Definition at line 121 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.397 std::binary\_negate< \_Predicate > Class Template Reference

One of the [negation functors](#).

Inheritance diagram for std::binary\_negate< \_Predicate >:



### Public Types

- typedef \_Predicate::first\_argument\_type [first\\_argument\\_type](#)
- typedef bool [result\\_type](#)
- typedef \_Predicate::second\_argument\_type [second\\_argument\\_type](#)

### Public Member Functions

- **binary\_negate** (const \_Predicate &\_\_x)
- bool **operator()** (const typename \_Predicate::first\_argument\_type &\_\_x, const typename \_Predicate::second\_argument\_type &\_\_y) const

### Protected Attributes

- \_Predicate \_M\_pred

### 5.397.1 Detailed Description

**template<typename \_Predicate> class std::binary\_negate< \_Predicate >**

One of the [negation functors](#).

Definition at line 375 of file stl\_function.h.

### 5.397.2 Member Typedef Documentation

- 5.397.2.1** typedef \_Predicate::first\_argument\_type std::binary\_function< \_Predicate::first\_argument\_type, \_Predicate::second\_argument\_type, bool >::first\_argument\_type **[inherited]**

`first_argument_type` is the type of the first argument

Definition at line 118 of file `stl_function.h`.

**5.397.2.2** `typedef bool std::binary_function<_Predicate::first_argument_type, _Predicate::second_argument_type, bool>::result_type [inherited]`

`result_type` is the return type

Definition at line 124 of file `stl_function.h`.

**5.397.2.3** `typedef _Predicate::second_argument_type std::binary_function<_Predicate::first_argument_type, _Predicate::second_argument_type, bool>::second_argument_type [inherited]`

`second_argument_type` is the type of the second argument

Definition at line 121 of file `stl_function.h`.

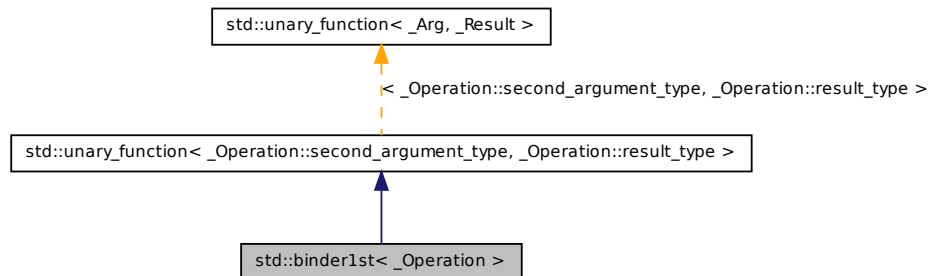
The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 5.398 `std::binder1st<_Operation>` Class Template Reference

One of the [binder functors](#).

Inheritance diagram for `std::binder1st<_Operation>`:



### Public Types

- typedef `_Operation::second_argument_type` [argument\\_type](#)
- typedef `_Operation::result_type` [result\\_type](#)

### Public Member Functions

- **binder1st** (`const _Operation &__x`, `const typename _Operation::first_argument_type &__y`)
- `_Operation::result_type` **operator()** (`typename _Operation::second_argument_type &__x`) `const`
- `_Operation::result_type` **operator()** (`const typename _Operation::second_argument_type &__x`) `const`

### Protected Attributes

- `_Operation` **op**
- `_Operation::first_argument_type` **value**

#### 5.398.1 Detailed Description

`template<typename _Operation> class std::binder1st<_Operation>`

One of the [binder functors](#).

Definition at line 100 of file `binders.h`.

### 5.398.2 Member Typedef Documentation

**5.398.2.1** `typedef _Operation::second_argument_type std::unary_function<  
_Operation::second_argument_type , _Operation::result_type  
>::argument_type [inherited]`

`argument_type` is the type of the argument

Definition at line 105 of file `stl_function.h`.

**5.398.2.2** `typedef _Operation::result_type std::unary_function<  
_Operation::second_argument_type , _Operation::result_type  
>::result_type [inherited]`

`result_type` is the return type

Definition at line 108 of file `stl_function.h`.

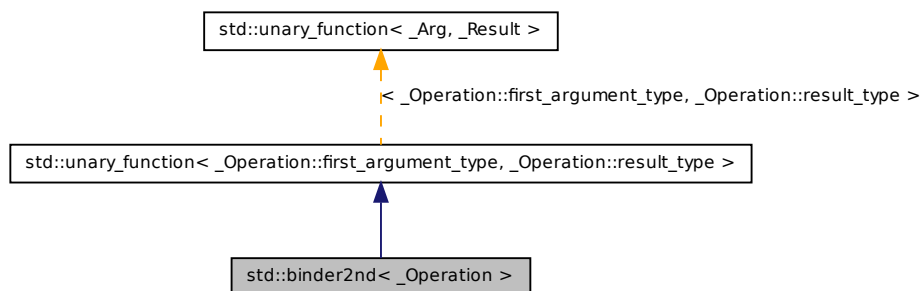
The documentation for this class was generated from the following file:

- [binders.h](#)

## 5.399 std::binder2nd< \_Operation > Class Template Reference

One of the [binder functors](#).

Inheritance diagram for `std::binder2nd< _Operation >`:



## Public Types

- typedef \_Operation::first\_argument\_type [argument\\_type](#)
- typedef \_Operation::result\_type [result\\_type](#)

## Public Member Functions

- **binder2nd** (const \_Operation &\_\_x, const typename \_Operation::second\_argument\_type &\_\_y)
- \_Operation::result\_type **operator()** (typename \_Operation::first\_argument\_type &\_\_x) const
- \_Operation::result\_type **operator()** (const typename \_Operation::first\_argument\_type &\_\_x) const

## Protected Attributes

- \_Operation **op**
- \_Operation::second\_argument\_type **value**

### 5.399.1 Detailed Description

**template<typename \_Operation> class std::binder2nd< \_Operation >**

One of the [binder functors](#).

Definition at line 135 of file binders.h.

### 5.399.2 Member Typedef Documentation

**5.399.2.1** typedef \_Operation::first\_argument\_type std::unary\_function< \_Operation::first\_argument\_type , \_Operation::result\_type >::argument\_type **[inherited]**

`argument_type` is the type of the argument

Definition at line 105 of file stl\_function.h.

**5.399.2.2** typedef \_Operation::result\_type std::unary\_function< \_Operation::first\_argument\_type , \_Operation::result\_type >::result\_type **[inherited]**



## 5.400 `std::binomial_distribution< _IntType >` Class Template Reference 2374

`result_type` is the return type

Definition at line 108 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [binders.h](#)

## 5.400 `std::binomial_distribution< _IntType >` Class Template Reference

A discrete binomial random number distribution.

### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_IntType` [result\\_type](#)

### Public Member Functions

- **`binomial_distribution`** (`_IntType __t=_IntType(1), double __p=0.5`)
- **`binomial_distribution`** (`const param\_type &__p`)
- [result\\_type](#) **`max`** () const
- [result\\_type](#) **`min`** () const
- `template<typename _UniformRandomNumberGenerator >`  
[result\\_type](#) **`operator()`** (`_UniformRandomNumberGenerator &__urng, const param\_type &__p`)
- `template<typename _UniformRandomNumberGenerator >`  
[result\\_type](#) **`operator()`** (`_UniformRandomNumberGenerator &__urng`)
- `double p` () const
- `void param` (`const param\_type &__param`)
- [param\\_type](#) **`param`** () const
- `void reset` ()
- `_IntType t` () const

### Friends

- `template<typename _IntType1, typename _CharT, typename _Traits >`  
`std::basic\_ostream< _CharT, _Traits > & operator<< (std::basic\_ostream< _CharT, _Traits > &, const std::binomial\_distribution< _IntType1 > &)`

- `template<typename _IntType1 >`  
`bool operator==(const std::binomial\_distribution< _IntType1 > &__d1, const`  
`std::binomial\_distribution< _IntType1 > &__d2)`
- `template<typename _IntType1, typename _CharT, typename _Traits >`  
`std::basic\_istream< _CharT, _Traits > & operator>> (std::basic\_istream< _`  
`CharT, _Traits > &, std::binomial\_distribution< _IntType1 > &)`

### 5.400.1 Detailed Description

`template<typename _IntType = int> class std::binomial\_distribution< _IntType >`

A discrete binomial random number distribution. The formula for the binomial probability density function is  $p(i|t, p) = \binom{n}{i} p^i (1 - p)^{t-i}$  where  $t$  and  $p$  are the parameters of the distribution.

Definition at line 3394 of file `random.h`.

### 5.400.2 Member Typedef Documentation

**5.400.2.1** `template<typename _IntType = int> typedef _IntType`  
`std::binomial\_distribution< _IntType >::result_type`

The type of the range of the distribution.

Definition at line 3401 of file `random.h`.

### 5.400.3 Member Function Documentation

**5.400.3.1** `template<typename _IntType = int> result_type`  
`std::binomial\_distribution< _IntType >::max ( ) const \[inline\]`

Returns the least upper bound value of the distribution.

Definition at line 3504 of file `random.h`.

Referenced by `std::binomial\_distribution< _IntType >::operator()()`.

**5.400.3.2** `template<typename _IntType = int> result_type`  
`std::binomial\_distribution< _IntType >::min ( ) const \[inline\]`

Returns the greatest lower bound value of the distribution.

Definition at line 3497 of file random.h.

**5.400.3.3** `template<typename _IntType = int> template<typename  
_UniformRandomNumberGenerator > result_type  
std::binomial_distribution< _IntType >::operator() (  
_UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 3512 of file random.h.

References `std::binomial_distribution< _IntType >::operator()()`, and `std::binomial_distribution< _IntType >::param()`.

Referenced by `std::binomial_distribution< _IntType >::operator()()`.

**5.400.3.4** `template<typename _IntType > template<typename  
_UniformRandomNumberGenerator > binomial_distribution<  
_IntType >::result_type std::binomial_distribution< _IntType  
>::operator() ( _UniformRandomNumberGenerator & __urng,  
const param_type & __param )`

A rejection algorithm when  $t * p \geq 8$  and a simple waiting time method - the second in the referenced book - otherwise. NB: The former is available only if `_GLIBCXX_USE_C99_MATH_TR1` is defined.

Reference: Devroye, L. Non-Uniform Random Variates Generation. Springer-Verlag, New York, 1986, Ch. X, Sect. 4 (+ Errata!).

Definition at line 1432 of file random.tcc.

References `std::abs()`, `std::log()`, and `std::binomial_distribution< _IntType >::max()`.

**5.400.3.5** `template<typename _IntType = int> double  
std::binomial_distribution< _IntType >::p ( ) const [inline]`

Returns the distribution `p` parameter.

Definition at line 3475 of file random.h.

**5.400.3.6** `template<typename _IntType = int> void std::binomial_distribution<_IntType>::param ( const param_type & __param )  
[inline]`

Sets the parameter set of the distribution.

**Parameters**

`__param` The new parameter set of the distribution.

Definition at line 3490 of file random.h.

**5.400.3.7** `template<typename _IntType = int> param_type  
std::binomial_distribution<_IntType>::param ( ) const  
[inline]`

Returns the parameter set of the distribution.

Definition at line 3482 of file random.h.

Referenced by `std::binomial_distribution<_IntType>::operator()()`.

**5.400.3.8** `template<typename _IntType = int> void  
std::binomial_distribution<_IntType>::reset ( ) [inline]`

Resets the distribution state.

Definition at line 3461 of file random.h.

References `std::normal_distribution<_RealType>::reset()`.

**5.400.3.9** `template<typename _IntType = int> _IntType  
std::binomial_distribution<_IntType>::t ( ) const [inline]`

Returns the distribution  $t$  parameter.

Definition at line 3468 of file random.h.

#### 5.400.4 Friends And Related Function Documentation

**5.400.4.1** `template<typename _IntType = int> template<typename _IntType1 ,  
typename _CharT , typename _Traits > std::basic_ostream<_CharT,  
_Traits>& operator<< ( std::basic_ostream<_CharT, _Traits > & ,  
const std::binomial_distribution<_IntType1 > & ) [friend]`

Inserts a `binomial_distribution` random number distribution `__x` into the output stream `__os`.

##### Parameters

`__os` An output stream.  
`__x` A `binomial_distribution` random number distribution.

##### Returns

The output stream with the state of `__x` inserted or in an error state.

**5.400.4.2** `template<typename _IntType = int> template<typename _IntType1  
> bool operator==( const std::binomial_distribution<_IntType1 >  
& __d1, const std::binomial_distribution<_IntType1 > & __d2 )  
[friend]`

Return true if two binomial distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 3527 of file `random.h`.

**5.400.4.3** `template<typename _IntType = int> template<typename _IntType1 ,  
typename _CharT , typename _Traits > std::basic_istream<_CharT,  
_Traits>& operator>> ( std::basic_istream<_CharT, _Traits > & ,  
std::binomial_distribution<_IntType1 > & ) [friend]`

Extracts a `binomial_distribution` random number distribution `__x` from the input stream `__is`.

##### Parameters

`__is` An input stream.  
`__x` A `binomial_distribution` random number generator engine.

## 5.401 `std::binomial_distribution<_IntType>::param_type` Struct Reference 2379

### Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.401 `std::binomial_distribution<_IntType>::param_type` Struct Reference

### Public Types

- typedef [binomial\\_distribution<\\_IntType>](#) **distribution\_type**

### Public Member Functions

- **param\_type** (`_IntType __t=_IntType(1), double __p=0.5`)
- double **p** () const
- `_IntType t` () const

### Friends

- class **binomial\_distribution<\_IntType>**
- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

#### 5.401.1 Detailed Description

`template<typename _IntType = int> struct std::binomial_distribution<_IntType>::param_type`

Parameter type.

Definition at line 3403 of file `random.h`.

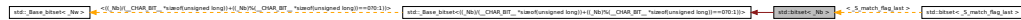
The documentation for this struct was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.402 std::bitset<\_Nb> Class Template Reference

The bitset class represents a *fixed-size* sequence of bits.

Inheritance diagram for std::bitset<\_Nb>:



### Classes

- class [reference](#)

### Public Member Functions

- constexpr [bitset](#) ()
- constexpr [bitset](#) (unsigned long long \_\_val)
- template<class \_CharT, class \_Traits, class \_Alloc >  
[bitset](#) (const [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_s, size\_t \_\_position, size\_t \_\_n)
- template<class \_CharT, class \_Traits, class \_Alloc >  
[bitset](#) (const [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_s, size\_t \_\_position, size\_t \_\_n, \_CharT \_\_zero, \_CharT \_\_one=\_CharT('1'))
- template<class \_CharT, class \_Traits, class \_Alloc >  
[bitset](#) (const [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_s, size\_t \_\_position=0)
- template<typename \_CharT >  
[bitset](#) (const \_CharT \*\_\_str, typename [std::basic\\_string](#)< \_CharT >::size\_type \_\_n=[std::basic\\_string](#)< \_CharT >::npos, \_CharT \_\_zero=\_CharT('0'), \_CharT \_\_one=\_CharT('1'))
- size\_t [\\_Find\\_first](#) () const
- size\_t [\\_Find\\_next](#) (size\_t \_\_prev) const
- template<class \_CharT, class \_Traits >  
void [\\_M\\_copy\\_from\\_ptr](#) (const \_CharT \*, size\_t, size\_t, size\_t, \_CharT, \_CharT)
- template<class \_CharT, class \_Traits, class \_Alloc >  
void [\\_M\\_copy\\_from\\_string](#) (const [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_s, size\_t \_\_pos, size\_t \_\_n, \_CharT \_\_zero, \_CharT \_\_one)
- template<class \_CharT, class \_Traits, class \_Alloc >  
void [\\_M\\_copy\\_from\\_string](#) (const [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_s, size\_t \_\_pos, size\_t \_\_n)
- template<class \_CharT, class \_Traits, class \_Alloc >  
void [\\_M\\_copy\\_to\\_string](#) ([std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &, \_CharT, \_CharT) const

- `template<class _CharT, class _Traits, class _Alloc >`  
`void _M_copy_to_string (std::basic_string< _CharT, _Traits, _Alloc > &__s)`  
`const`
- `bool all () const`
- `bool any () const`
- `size_t count () const`
- `bitset<_Nb> & flip ()`
- `bitset<_Nb> & flip (size_t __position)`
- `bool none () const`
- `bitset<_Nb> operator~ () const`
- `bitset<_Nb> & reset ()`
- `bitset<_Nb> & reset (size_t __position)`
- `bitset<_Nb> & set ()`
- `bitset<_Nb> & set (size_t __position, bool __val=true)`
- `constexpr size_t size () const`
- `bool test (size_t __position) const`
- `template<class _CharT, class _Traits >`  
`std::basic_string< _CharT, _Traits, std::allocator< _CharT > > to_string ()`  
`const`
- `template<class _CharT >`  
`std::basic_string< _CharT, std::char_traits< _CharT >, std::allocator< _CharT`  
`> > to_string () const`
- `template<class _CharT >`  
`std::basic_string< _CharT, std::char_traits< _CharT >, std::allocator< _CharT`  
`> > to_string (_CharT __zero, _CharT __one=_CharT('1')) const`
- `template<class _CharT, class _Traits, class _Alloc >`  
`std::basic_string< _CharT, _Traits, _Alloc > to_string () const`
- `template<class _CharT, class _Traits >`  
`std::basic_string< _CharT, _Traits, std::allocator< _CharT > > to_string (_-`  
`CharT __zero, _CharT __one=_CharT('1')) const`
- `template<class _CharT, class _Traits, class _Alloc >`  
`std::basic_string< _CharT, _Traits, _Alloc > to_string (_CharT __zero, _CharT`  
`__one=_CharT('1')) const`
- `std::basic_string< char, std::char_traits< char >, std::allocator< char > > to_-`  
`string (char __zero, char __one= '1') const`
- `std::basic_string< char, std::char_traits< char >, std::allocator< char > > to_-`  
`string () const`
- `unsigned long long to_ullong () const`
- `unsigned long to_ulong () const`
  
- `bitset<_Nb> & operator&= (const bitset<_Nb> &__rhs)`
- `bitset<_Nb> & operator|= (const bitset<_Nb> &__rhs)`
- `bitset<_Nb> & operator^= (const bitset<_Nb> &__rhs)`



- `bitset<_Nb> & operator<<=` (size\_t \_\_position)
- `bitset<_Nb> & operator>>=` (size\_t \_\_position)
- `bitset<_Nb> & _Unchecked_set` (size\_t \_\_pos)
- `bitset<_Nb> & _Unchecked_set` (size\_t \_\_pos, int \_\_val)
- `bitset<_Nb> & _Unchecked_reset` (size\_t \_\_pos)
- `bitset<_Nb> & _Unchecked_flip` (size\_t \_\_pos)
- `bool _Unchecked_test` (size\_t \_\_pos) const
- `reference operator[ ]` (size\_t \_\_position)
- `bool operator[ ]` (size\_t \_\_position) const
- `bool operator==` (const `bitset<_Nb>` & \_\_rhs) const
- `bool operator!=` (const `bitset<_Nb>` & \_\_rhs) const
- `bitset<_Nb> operator<<` (size\_t \_\_position) const
- `bitset<_Nb> operator>>` (size\_t \_\_position) const

### Private Member Functions

- `size_t _M_are_all_aux` () const
- `void _M_do_and` (const `_Base_bitset<_Nw>` & \_\_x)
- `size_t _M_do_count` () const
- `size_t _M_do_find_first` (size\_t \_\_not\_found) const
- `size_t _M_do_find_next` (size\_t \_\_prev, size\_t \_\_not\_found) const
- `void _M_do_flip` ()
- `void _M_do_left_shift` (size\_t \_\_shift)
- `void _M_do_or` (const `_Base_bitset<_Nw>` & \_\_x)
- `void _M_do_reset` ()
- `void _M_do_right_shift` (size\_t \_\_shift)
- `void _M_do_set` ()
- `unsigned long long _M_do_to_ullong` () const
- `unsigned long _M_do_to_ulong` () const
- `void _M_do_xor` (const `_Base_bitset<_Nw>` & \_\_x)
- `const _WordT * _M_getdata` () const
- `_WordT & _M_getword` (size\_t \_\_pos)
- `_WordT _M_getword` (size\_t \_\_pos) const
- `_WordT & _M_hiword` ()
- `constexpr _WordT _M_hiword` () const
- `bool _M_is_any` () const
- `bool _M_is_equal` (const `_Base_bitset<_Nw>` & \_\_x) const

**Static Private Member Functions**

- static constexpr `_WordT _S_maskbit` (`size_t __pos`)
- static constexpr `size_t _S_whichbit` (`size_t __pos`)
- static constexpr `size_t _S_whichbyte` (`size_t __pos`)
- static constexpr `size_t _S_whichword` (`size_t __pos`)

**Private Attributes**

- `_WordT _M_w` [`_Nw`]

**Friends**

- class **hash**
- class **reference**

**5.402.1 Detailed Description**

`template<size_t _Nb> class std::bitset<_Nb>`

The `bitset` class represents a *fixed-size* sequence of bits. (Note that `bitset` does *not* meet the formal requirements of a `container`. Mainly, it lacks iterators.)

The template argument, *Nb*, may be any non-negative number, specifying the number of bits (e.g., "0", "12", "1024\*1024").

In the general unoptimized case, storage is allocated in word-sized blocks. Let *B* be the number of bits in a word, then  $(Nb+(B-1))/B$  words will be used for storage. *B* - *NbB* bits are unused. (They are the high-order bits in the highest word.) It is a class invariant that those unused bits are always zero.

If you think of `bitset` as a *simple array of bits*, be aware that your mental picture is reversed: a `bitset` behaves the same way as bits in integers do, with the bit at index 0 in the *least significant / right-hand* position, and the bit at index *Nb*-1 in the *most significant / left-hand* position. Thus, unlike other containers, a `bitset`'s index *counts from right to left*, to put it very loosely.

This behavior is preserved when translating to and from strings. For example, the first line of the following program probably prints *b('a') is 0001100001* on a modern ASCII system.

```
#include <bitset>
#include <iostream>
#include <sstream>

using namespace std;
```

```

int main()
{
 long a = 'a';
 bitset<10> b(a);

 cout << "b('a') is " << b << endl;

 ostringstream s;
 s << b;
 string str = s.str();
 cout << "index 3 in the string is " << str[3] << " but\n"
 << "index 3 in the bitset is " << b[3] << endl;
}

```

Also see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt12ch33s02.html> for a description of extensions.

Most of the actual code isn't contained in `bitset<>` itself, but in the base class `_Base_bitset`. The base class works with whole words, not with individual bits. This allows us to specialize `_Base_bitset` for the important special case where the bitset is only a single word.

Extra confusion can result due to the fact that the storage for `_Base_bitset` is a regular array, and is indexed as such. This is carefully encapsulated.

Definition at line 722 of file `bitset`.

### 5.402.2 Constructor & Destructor Documentation

#### 5.402.2.1 `template<size_t _Nb> constexpr std::bitset<_Nb>::bitset ( )` [`inline`]

All bits set to zero.

Definition at line 816 of file `bitset`.

#### 5.402.2.2 `template<size_t _Nb> constexpr std::bitset<_Nb>::bitset (` `unsigned long long __val )` [`inline`]

Initial bits bitwise-copied from a single word (others set to zero).

Definition at line 821 of file `bitset`.

**5.402.2.3** `template<size_t _Nb> template<class _CharT, class _Traits, class _Alloc> std::bitset<_Nb>::bitset ( const std::basic_string<_CharT, _Traits, _Alloc> & __s, size_t __position = 0 )`  
`[inline, explicit]`

Use a subset of a string.

#### Parameters

*s* A string of 0 and 1 characters.

*position* Index of the first character in *s* to use; defaults to zero.

#### Exceptions

*std::out\_of\_range* If *pos* is bigger the size of *s*.

*std::invalid\_argument* If a character appears in the string which is neither 0 nor 1.

Definition at line 840 of file `bitset`.

**5.402.2.4** `template<size_t _Nb> template<class _CharT, class _Traits, class _Alloc> std::bitset<_Nb>::bitset ( const std::basic_string<_CharT, _Traits, _Alloc> & __s, size_t __position, size_t __n )`  
`[inline]`

Use a subset of a string.

#### Parameters

*s* A string of 0 and 1 characters.

*position* Index of the first character in *s* to use.

*n* The number of characters to copy.

#### Exceptions

*std::out\_of\_range* If *pos* is bigger the size of *s*.

*std::invalid\_argument* If a character appears in the string which is neither 0 nor 1.

Definition at line 862 of file `bitset`.

**5.402.2.5** `template<size_t _Nb> template<typename _CharT> std::bitset<_Nb>::bitset ( const _CharT * __str, typename std::basic_string<_CharT>::size_type __n = std::basic_string<_CharT>::npos, _CharT __zero = _CharT('0'), _CharT __one = _CharT('1') ) [inline, explicit]`

Construct from a character array.

#### Parameters

- str* An array of characters *zero* and *one*.
- n* The number of characters to use.
- zero* The character corresponding to the value 0.
- one* The character corresponding to the value 1.

#### Exceptions

- std::invalid\_argument* If a character appears in the string which is neither *zero* nor *one*.

Definition at line 898 of file `bitset`.

### 5.402.3 Member Function Documentation

**5.402.3.1** `template<size_t _Nb> bool std::bitset<_Nb>::all ( ) const [inline]`

Tests whether all the bits are on.

#### Returns

- True if all the bits are set.

Definition at line 1290 of file `bitset`.

**5.402.3.2** `template<size_t _Nb> bool std::bitset<_Nb>::any ( ) const [inline]`

Tests whether any of the bits are on.

**Returns**

True if at least one bit is set.

Definition at line 1298 of file `bitset`.

**5.402.3.3** `template<size_t _Nb> size_t std::bitset<_Nb>::count ( ) const`  
`[inline]`

Returns the number of bits which are set.

Definition at line 1250 of file `bitset`.

**5.402.3.4** `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb>::flip (`  
`size_t __position ) [inline]`

Toggles a given bit to its opposite value.

**Parameters**

*position* The index of the bit.

**Exceptions**

*std::out\_of\_range* If *pos* is bigger the size of the set.

Definition at line 1090 of file `bitset`.

**5.402.3.5** `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb>::flip ( )`  
`[inline]`

Toggles every bit to its opposite value.

Definition at line 1077 of file `bitset`.

**5.402.3.6** `template<size_t _Nb> bool std::bitset<_Nb>::none ( ) const`  
`[inline]`

Tests whether any of the bits are on.

**Returns**

True if none of the bits are set.

Definition at line 1306 of file `bitset`.

**5.402.3.7** `template<size_t _Nb> bool std::bitset<_Nb>::operator!=( const  
bitset<_Nb> & __rhs ) const [inline]`

These comparisons for equality/inequality are, well, *bitwise*.

Definition at line 1265 of file `bitset`.

**5.402.3.8** `template<size_t _Nb> bitset<_Nb> & std::bitset<_Nb  
>::operator&= ( const bitset<_Nb> & __rhs ) [inline]`

Operations on bitsets.

**Parameters**

*rhs* A same-sized `bitset`.

These should be self-explanatory.

Definition at line 924 of file `bitset`.

**5.402.3.9** `template<size_t _Nb> bitset<_Nb> std::bitset<_Nb  
>::operator<< ( size_t __position ) const [inline]`

Self-explanatory.

Definition at line 1312 of file `bitset`.

**5.402.3.10** `template<size_t _Nb> bitset<_Nb> & std::bitset<_Nb  
>::operator<<= ( size_t __position ) [inline]`

Operations on bitsets.

**Parameters**

*position* The number of places to shift.

These should be self-explanatory.

Definition at line 953 of file `bitset`.

**5.402.3.11** `template<size_t _Nb> bool std::bitset<_Nb>::operator==( const  
bitset<_Nb> & __rhs ) const [inline]`

These comparisons for equality/inequality are, well, *bitwise*.

Definition at line 1261 of file `bitset`.

**5.402.3.12** `template<size_t _Nb> bitset<_Nb> std::bitset<_Nb  
>::operator>>( size_t __position ) const [inline]`

Self-explanatory.

Definition at line 1316 of file `bitset`.

**5.402.3.13** `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb  
>::operator>>=( size_t __position ) [inline]`

Operations on bitsets.

#### Parameters

*position* The number of places to shift.

These should be self-explanatory.

Definition at line 966 of file `bitset`.

**5.402.3.14** `template<size_t _Nb> bool std::bitset<_Nb>::operator[]( size_t  
__position ) const [inline]`

Array-indexing support.

#### Parameters

*position* Index into the `bitset`.



**Returns**

A bool for a *const bitset*. For non-const bitsets, an instance of the reference proxy class.

**Note**

These operators do no range checking and throw no exceptions, as required by DR 11 to the standard.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` Note that this implementation already resolves DR 11 (items 1 and 2), but does not do the range-checking required by that DR's resolution. -pme The DR has since been changed: range-checking is a precondition (users' responsibility), and these functions must not throw. -pme

Definition at line 1122 of file `bitset`.

**5.402.3.15** `template<size_t _Nb> reference std::bitset<_Nb>::operator[] (size_t __position) [inline]`

Array-indexing support.

**Parameters**

*position* Index into the bitset.

**Returns**

A bool for a *const bitset*. For non-const bitsets, an instance of the reference proxy class.

**Note**

These operators do no range checking and throw no exceptions, as required by DR 11 to the standard.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` Note that this implementation already resolves DR 11 (items 1 and 2), but does not do the range-checking required by that DR's resolution. -pme The DR has since been changed: range-checking is a precondition (users' responsibility), and these functions must not throw. -pme

Definition at line 1118 of file `bitset`.

**5.402.3.16** `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb>::operator^= (const bitset<_Nb> & __rhs) [inline]`

Operations on bitsets.

#### Parameters

*rhs* A same-sized bitset.

These should be self-explanatory.

Definition at line 938 of file bitset.

**5.402.3.17** `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb>::operator|= ( const bitset<_Nb> & __rhs ) [inline]`

Operations on bitsets.

#### Parameters

*rhs* A same-sized bitset.

These should be self-explanatory.

Definition at line 931 of file bitset.

**5.402.3.18** `template<size_t _Nb> bitset<_Nb> std::bitset<_Nb>::operator~ ( ) const [inline]`

See the no-argument [flip\(\)](#).

Definition at line 1099 of file bitset.

**5.402.3.19** `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb>::reset ( ) [inline]`

Sets every bit to false.

Definition at line 1052 of file bitset.

**5.402.3.20** `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb>::reset ( size_t __position ) [inline]`

Sets a given bit to false.

#### Parameters

*position* The index of the bit.

#### Exceptions

*std::out\_of\_range* If *pos* is bigger the size of the set.

Same as writing `set(pos, false)`.

Definition at line 1066 of file `bitset`.

**5.402.3.21** `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb>::set ( )`  
`[inline]`

Sets every bit to true.

Definition at line 1027 of file `bitset`.

**5.402.3.22** `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb>::set (`  
`size_t __position, bool __val = true ) [inline]`

Sets a given bit to a particular value.

#### Parameters

*position* The index of the bit.

*val* Either true or false, defaults to true.

#### Exceptions

*std::out\_of\_range* If *pos* is bigger the size of the set.

Definition at line 1041 of file `bitset`.

**5.402.3.23** `template<size_t _Nb> constexpr size_t std::bitset<_Nb>::size ( )`  
`const [inline]`

Returns the total number of bits.

Definition at line 1255 of file `bitset`.

**5.402.3.24** `template<size_t _Nb> bool std::bitset<_Nb>::test ( size_t  
__position ) const [inline]`

Tests the value of a bit.

#### Parameters

*position* The index of a bit.

#### Returns

The value at *pos*.

#### Exceptions

[\*std::out\\_of\\_range\*](#) If *pos* is bigger the size of the set.

Definition at line 1276 of file `bitset`.

**5.402.3.25** `template<size_t _Nb> template<class _CharT, class _Traits, class  
_Alloc> std::basic_string<_CharT, _Traits, _Alloc> std::bitset<  
_Nb>::to_string ( ) const [inline]`

Returns a character interpretation of the bitset.

#### Returns

The string equivalent of the bits.

Note the ordering of the bits: decreasing character positions correspond to increasing bit positions (see the main class notes for an example).

Definition at line 1152 of file `bitset`.

**5.402.3.26** `template<size_t _Nb> unsigned long std::bitset<_Nb>::to_ulong (  
) const [inline]`

Returns a numerical interpretation of the bitset.

#### Returns

The integral equivalent of the bits.

### Exceptions

*`std::overflow_error`* If there are too many bits to be represented in an unsigned long.

Definition at line 1133 of file `bitset`.

The documentation for this class was generated from the following file:

- `bitset`

## 5.403 `std::bitset<_Nb>::reference` Class Reference

### Public Member Functions

- `reference` (`bitset` &\_\_b, `size_t` \_\_pos)
- `reference` & `flip` ()
- `operator bool` () const
- `reference` & `operator=` (const `reference` &\_\_j)
- `reference` & `operator=` (`bool` \_\_x)
- `bool operator~` () const

### Friends

- class `bitset`

#### 5.403.1 Detailed Description

`template<size_t _Nb> class std::bitset<_Nb>::reference`

This encapsulates the concept of a single bit. An instance of this class is a proxy for an actual bit; this way the individual bit operations are done as faster word-size bitwise instructions.

Most users will never need to use this class directly; conversions to and from `bool` are automatic and should be transparent. Overloaded operators help to preserve the illusion.

(On a typical system, this *bit reference* is 64 times the size of an actual bit. Ha.)

Definition at line 753 of file `bitset`.

The documentation for this class was generated from the following file:

- `bitset`

## 5.404 `std::cauchy_distribution< _RealType >` Class Template Reference

A [`cauchy\_distribution`](#) random number distribution.

### Classes

- struct [`param\_type`](#)

### Public Types

- typedef `_RealType` [`result\_type`](#)

### Public Member Functions

- **`cauchy_distribution`** (`_RealType` `__a`=`_RealType(0)`, `_RealType` `__b`=`_RealType(1)`)
- **`cauchy_distribution`** (const [`param\_type`](#) &`__p`)
- `_RealType` **`a`** () const
- `_RealType` **`b`** () const
- [`result\_type`](#) **`max`** () const
- [`result\_type`](#) **`min`** () const
- template<typename `_UniformRandomNumberGenerator` >  
[`result\_type`](#) **`operator()`** (`_UniformRandomNumberGenerator` &`__urng`, const [`param\_type`](#) &`__p`)
- template<typename `_UniformRandomNumberGenerator` >  
[`result\_type`](#) **`operator()`** (`_UniformRandomNumberGenerator` &`__urng`)
- void [`param`](#) (const [`param\_type`](#) &`__param`)
- [`param\_type`](#) [`param`](#) () const
- void [`reset`](#) ()

#### 5.404.1 Detailed Description

**template<typename `_RealType` = `double`> class `std::cauchy_distribution< _RealType >`**

A [`cauchy\_distribution`](#) random number distribution. The formula for the normal probability mass function is  $p(x|a, b) = (\pi b (1 + (\frac{x-a}{b})^2))^{-1}$

Definition at line 2681 of file `random.h`.

### **5.404.2 Member Typedef Documentation**

**5.404.2.1** `template<typename _RealType = double> typedef _RealType  
std::cauchy_distribution< _RealType >::result_type`

The type of the range of the distribution.

Definition at line 2688 of file random.h.

### **5.404.3 Member Function Documentation**

**5.404.3.1** `template<typename _RealType = double> result_type  
std::cauchy_distribution< _RealType >::max ( ) const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 2772 of file random.h.

**5.404.3.2** `template<typename _RealType = double> result_type  
std::cauchy_distribution< _RealType >::min ( ) const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 2765 of file random.h.

**5.404.3.3** `template<typename _RealType = double> template<typename  
_UniformRandomNumberGenerator > result_type  
std::cauchy_distribution< _RealType >::operator() (  
_UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 2780 of file random.h.

References `std::cauchy_distribution< _RealType >::operator()()`, and `std::cauchy_distribution< _RealType >::param()`.

Referenced by `std::cauchy_distribution< _RealType >::operator()()`.

## 5.405 `std::cauchy_distribution<_RealType>::param_type` Struct Reference 2397

**5.404.3.4** `template<typename _RealType = double> void  
std::cauchy_distribution<_RealType>::param ( const param_type  
& __param ) [inline]`

Sets the parameter set of the distribution.

### Parameters

`__param` The new parameter set of the distribution.

Definition at line 2758 of file `random.h`.

**5.404.3.5** `template<typename _RealType = double> param_type  
std::cauchy_distribution<_RealType>::param ( ) const  
[inline]`

Returns the parameter set of the distribution.

Definition at line 2750 of file `random.h`.

Referenced by `std::cauchy_distribution<_RealType>::operator()()`, `std::operator==( )`, and `std::operator>>()`.

**5.404.3.6** `template<typename _RealType = double> void  
std::cauchy_distribution<_RealType>::reset ( ) [inline]`

Resets the distribution state.

Definition at line 2732 of file `random.h`.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.405 `std::cauchy_distribution<_RealType>::param_type` Struct Reference

### Public Types

- typedef `cauchy_distribution<_RealType>` `distribution_type`



**Public Member Functions**

- **param\_type** (\_RealType \_\_a=\_RealType(0), \_RealType \_\_b=\_RealType(1))
- \_RealType **a** () const
- \_RealType **b** () const

**Friends**

- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

**5.405.1 Detailed Description**

**template<typename \_RealType = double> struct std::cauchy\_distribution< \_RealType >::param\_type**

Parameter type.

Definition at line 2690 of file random.h.

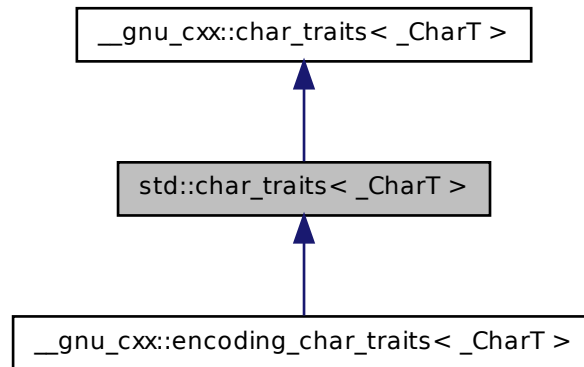
The documentation for this struct was generated from the following file:

- [random.h](#)

**5.406 `std::char_traits<_CharT>` Struct Template Reference**

Basis for explicit traits specializations.

Inheritance diagram for std::char\_traits<\_CharT>:



### Public Types

- typedef `_CharT` **char\_type**
- typedef `_Char_types<_CharT>::int_type` **int\_type**
- typedef `_Char_types<_CharT>::off_type` **off\_type**
- typedef `_Char_types<_CharT>::pos_type` **pos\_type**
- typedef `_Char_types<_CharT>::state_type` **state\_type**

### Static Public Member Functions

- static void **assign** (`char_type` &\_\_c1, const `char_type` &\_\_c2)
- static `char_type` \* **assign** (`char_type` \*\_\_s, std::size\_t \_\_n, `char_type` \_\_a)
- static int **compare** (const `char_type` \*\_\_s1, const `char_type` \*\_\_s2, std::size\_t \_\_n)
- static `char_type` \* **copy** (`char_type` \*\_\_s1, const `char_type` \*\_\_s2, std::size\_t \_\_n)
- static constexpr int\_type **eof** ()
- static constexpr bool **eq** (const `char_type` &\_\_c1, const `char_type` &\_\_c2)
- static constexpr bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2)
- static const `char_type` \* **find** (const `char_type` \*\_\_s, std::size\_t \_\_n, const `char_type` &\_\_a)

## 5.407 `std::char_traits< __gnu_cxx::character< V, I, S > >` Struct Template Reference 2400

---

- static `std::size_t length` (`const char_type *__s`)
- static constexpr `bool lt` (`const char_type &__c1, const char_type &__c2`)
- static `char_type * move` (`char_type *__s1, const char_type *__s2, std::size_t __n`)
- static constexpr `int_type not_eof` (`const int_type &__c`)
- static constexpr `char_type to_char_type` (`const int_type &__c`)
- static constexpr `int_type to_int_type` (`const char_type &__c`)

### 5.406.1 Detailed Description

`template<class _CharT> struct std::char_traits< _CharT >`

Basis for explicit traits specializations.

#### Note

For any given actual character type, this definition is probably wrong. Since this is just a thin wrapper around `__gnu_cxx::char_traits`, it is possible to achieve a more appropriate definition by specializing `__gnu_cxx::char_traits`.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt05ch13s03.html> for advice on how to make use of this class for *unusual* character types. Also, check out [include/ext/pod\\_char\\_traits.h](#).

Definition at line 229 of file `char_traits.h`.

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)

## 5.407 `std::char_traits< __gnu_cxx::character< V, I, S > >` Struct Template Reference

`char_traits<__gnu_cxx::character>` specialization.

#### Public Types

- typedef `__gnu_cxx::character< V, I, S > char_type`
- typedef `char_type::int_type int_type`
- typedef `streamoff off_type`
- typedef `fpos< state_type > pos_type`
- typedef `char_type::state_type state_type`

**Static Public Member Functions**

- static void **assign** ([char\\_type](#) &\_\_c1, const [char\\_type](#) &\_\_c2)
- static [char\\_type](#) \* **assign** ([char\\_type](#) \*\_\_s, size\_t \_\_n, [char\\_type](#) \_\_a)
- static int **compare** (const [char\\_type](#) \*\_\_s1, const [char\\_type](#) \*\_\_s2, size\_t \_\_n)
- static [char\\_type](#) \* **copy** ([char\\_type](#) \*\_\_s1, const [char\\_type](#) \*\_\_s2, size\_t \_\_n)
- static int\_type **eof** ()
- static bool **eq** (const [char\\_type](#) &\_\_c1, const [char\\_type](#) &\_\_c2)
- static bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2)
- static const [char\\_type](#) \* **find** (const [char\\_type](#) \*\_\_s, size\_t \_\_n, const [char\\_type](#) &\_\_a)
- static size\_t **length** (const [char\\_type](#) \*\_\_s)
- static bool **lt** (const [char\\_type](#) &\_\_c1, const [char\\_type](#) &\_\_c2)
- static [char\\_type](#) \* **move** ([char\\_type](#) \*\_\_s1, const [char\\_type](#) \*\_\_s2, size\_t \_\_n)
- static int\_type **not\_eof** (const int\_type &\_\_c)
- static [char\\_type](#) **to\_char\_type** (const int\_type &\_\_i)
- static int\_type **to\_int\_type** (const [char\\_type](#) &\_\_c)

**5.407.1 Detailed Description**

`template<typename V, typename I, typename S> struct std::char_traits< __gnu_cxx::character< V, I, S > >`

`char_traits<__gnu_cxx::character>` specialization.

Definition at line 93 of file `pod_char_traits.h`.

The documentation for this struct was generated from the following file:

- [pod\\_char\\_traits.h](#)

**5.408 `std::char_traits< char >` Struct Template Reference**

21.1.3.1 [char\\_traits](#) specializations

**Public Types**

- typedef char **char\_type**
- typedef int **int\_type**
- typedef [streamoff](#) **off\_type**
- typedef [streampos](#) **pos\_type**
- typedef mbstate\_t **state\_type**

**Static Public Member Functions**

- static void **assign** (char\_type &\_\_c1, const char\_type &\_\_c2)
- static char\_type \* **assign** (char\_type \*\_\_s, size\_t \_\_n, char\_type \_\_a)
- static int **compare** (const char\_type \*\_\_s1, const char\_type \*\_\_s2, size\_t \_\_n)
- static char\_type \* **copy** (char\_type \*\_\_s1, const char\_type \*\_\_s2, size\_t \_\_n)
- static constexpr int\_type **eof** ()
- static constexpr bool **eq** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2)
- static const char\_type \* **find** (const char\_type \*\_\_s, size\_t \_\_n, const char\_type &\_\_a)
- static size\_t **length** (const char\_type \*\_\_s)
- static constexpr bool **lt** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static char\_type \* **move** (char\_type \*\_\_s1, const char\_type \*\_\_s2, size\_t \_\_n)
- static constexpr int\_type **not\_eof** (const int\_type &\_\_c)
- static constexpr char\_type **to\_char\_type** (const int\_type &\_\_c)
- static constexpr int\_type **to\_int\_type** (const char\_type &\_\_c)

**5.408.1 Detailed Description**

`template<> struct std::char_traits< char >`

21.1.3.1 [char\\_traits](#) specializations

Definition at line 235 of file `char_traits.h`.

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)

**5.409 `std::char_traits< wchar_t >` Struct Template Reference**

21.1.3.2 [char\\_traits](#) specializations

**Public Types**

- typedef `wchar_t` **char\_type**
- typedef `wint_t` **int\_type**
- typedef [streamoff](#) **off\_type**
- typedef [wstreampos](#) **pos\_type**
- typedef `mbstate_t` **state\_type**

### Static Public Member Functions

- static void **assign** (char\_type &\_\_c1, const char\_type &\_\_c2)
- static char\_type \* **assign** (char\_type \*\_\_s, size\_t \_\_n, char\_type \_\_a)
- static int **compare** (const char\_type \*\_\_s1, const char\_type \*\_\_s2, size\_t \_\_n)
- static char\_type \* **copy** (char\_type \*\_\_s1, const char\_type \*\_\_s2, size\_t \_\_n)
- static constexpr int\_type **eof** ()
- static constexpr bool **eq** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2)
- static const char\_type \* **find** (const char\_type \*\_\_s, size\_t \_\_n, const char\_type &\_\_a)
- static size\_t **length** (const char\_type \*\_\_s)
- static constexpr bool **lt** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static char\_type \* **move** (char\_type \*\_\_s1, const char\_type \*\_\_s2, size\_t \_\_n)
- static constexpr int\_type **not\_eof** (const int\_type &\_\_c)
- static constexpr char\_type **to\_char\_type** (const int\_type &\_\_c)
- static constexpr int\_type **to\_int\_type** (const char\_type &\_\_c)

#### 5.409.1 Detailed Description

`template<> struct std::char_traits< wchar_t >`

21.1.3.2 [char\\_traits](#) specializations

Definition at line 306 of file `char_traits.h`.

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)

### 5.410 `std::chi_squared_distribution<_RealType>` Class Template Reference

A [chi\\_squared\\_distribution](#) random number distribution.

#### Classes

- struct [param\\_type](#)

#### Public Types

- typedef `_RealType` [result\\_type](#)

### Public Member Functions

- `chi_squared_distribution` (`_RealType __n=_RealType(1)`)
- `chi_squared_distribution` (`const param_type &__p`)
- `result_type max` () const
- `result_type min` () const
- `_RealType n` () const
- `template<typename _UniformRandomNumberGenerator >`  
`result_type operator()` (`_UniformRandomNumberGenerator &__urng`)
- `template<typename _UniformRandomNumberGenerator >`  
`result_type operator()` (`_UniformRandomNumberGenerator &__urng, const param_type &__p`)
- `void param` (`const param_type &__param`)
- `param_type param` () const
- `void reset` ()

### Friends

- `template<typename _RealType1, typename _CharT, typename _Traits >`  
`std::basic_ostream<_CharT, _Traits> & operator<<` (`std::basic_ostream<_CharT, _Traits> &, const std::chi_squared_distribution<_RealType1> &`)
- `template<typename _RealType1 >`  
`bool operator==` (`const std::chi_squared_distribution<_RealType1> &__d1, const std::chi_squared_distribution<_RealType1> &__d2`)
- `template<typename _RealType1, typename _CharT, typename _Traits >`  
`std::basic_istream<_CharT, _Traits> & operator>>` (`std::basic_istream<_CharT, _Traits> &, std::chi_squared_distribution<_RealType1> &`)

#### 5.410.1 Detailed Description

`template<typename _RealType = double> class std::chi_squared_distribution<_RealType>`

A `chi_squared_distribution` random number distribution. The formula for the normal probability mass function is  $p(x|n) = \frac{x^{(n/2)-1} e^{-x/2}}{\Gamma(n/2) 2^{n/2}}$

Definition at line 2516 of file `random.h`.

#### 5.410.2 Member Typedef Documentation

**5.410.2.1** `template<typename _RealType = double> typedef _RealType std::chi_squared_distribution<_RealType>::result_type`

The type of the range of the distribution.

## 5.410 `std::chi_squared_distribution<_RealType>` Class Template Reference 2405

Definition at line 2523 of file random.h.

### 5.410.3 Member Function Documentation

**5.410.3.1** `template<typename _RealType = double> result_type  
std::chi_squared_distribution<_RealType>::max ( ) const  
[inline]`

Returns the least upper bound value of the distribution.

Definition at line 2596 of file random.h.

**5.410.3.2** `template<typename _RealType = double> result_type  
std::chi_squared_distribution<_RealType>::min ( ) const  
[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 2589 of file random.h.

**5.410.3.3** `template<typename _RealType = double> template<typename  
_UniformRandomNumberGenerator> result_type  
std::chi_squared_distribution<_RealType>::operator() (   
_UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 2604 of file random.h.

**5.410.3.4** `template<typename _RealType = double> param_type  
std::chi_squared_distribution<_RealType>::param ( ) const  
[inline]`

Returns the parameter set of the distribution.

Definition at line 2574 of file random.h.



**5.410.3.5** `template<typename _RealType = double> void  
std::chi_squared_distribution<_RealType>::param ( const  
param_type & __param ) [inline]`

Sets the parameter set of the distribution.

#### Parameters

`__param` The new parameter set of the distribution.

Definition at line 2582 of file random.h.

**5.410.3.6** `template<typename _RealType = double> void  
std::chi_squared_distribution<_RealType>::reset ( ) [inline]`

Resets the distribution state.

Definition at line 2560 of file random.h.

References `std::gamma_distribution<_RealType>::reset()`.

#### 5.410.4 Friends And Related Function Documentation

**5.410.4.1** `template<typename _RealType = double> template<typename  
_RealType1 , typename _CharT , typename _Traits >  
std::basic_ostream<_CharT, _Traits>& operator<<  
( std::basic_ostream<_CharT, _Traits> &, const  
std::chi_squared_distribution<_RealType1> & ) [friend]`

Inserts a `chi_squared_distribution` random number distribution `__x` into the output stream `__os`.

#### Parameters

`__os` An output stream.

`__x` A `chi_squared_distribution` random number distribution.

#### Returns

The output stream with the state of `__x` inserted or in an error state.

5.410.4.2 `template<typename _RealType = double> template<typename _RealType1 > bool operator==( const std::chi_squared_distribution<_RealType1 > & __d1, const std::chi_squared_distribution<_RealType1 > & __d2 ) [friend]`

Return true if two Chi-squared distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 2624 of file random.h.

5.410.4.3 `template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename _Traits > std::basic_istream<_CharT, _Traits>& operator>>( std::basic_istream<_CharT, _Traits > & , std::chi_squared_distribution<_RealType1 > & ) [friend]`

Extracts a `chi_squared_distribution` random number distribution `__x` from the input stream `__is`.

#### Parameters

- `__is` An input stream.
- `__x` A `chi_squared_distribution` random number generator engine.

#### Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

## 5.411 `std::chi_squared_distribution<_RealType>::param_type` Struct Reference

### Public Types

- typedef [chi\\_squared\\_distribution](#)<\_RealType> `distribution_type`

### Public Member Functions

- `param_type` (`_RealType __n=_RealType(1)`)
- `_RealType n` () const

## 5.412 `std::chrono::duration< _Rep, _Period >` Struct Template Reference 2408

---

### Friends

- `bool operator==` (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

### 5.411.1 Detailed Description

`template<typename _RealType = double> struct std::chi_squared_distribution< _RealType >::param_type`

Parameter type.

Definition at line 2525 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.412 `std::chrono::duration< _Rep, _Period >` Struct Template Reference

`duration`

### Public Types

- `typedef _Period period`
- `typedef _Rep rep`

### Public Member Functions

- `constexpr duration` (const [duration](#) &)
- `template<typename _Rep2 , typename = typename enable_if<is_convertible<_Rep2, rep>::value && (treat_as_floating_point<rep>::value || !treat_as_floating_point<_Rep2>::value)>::type> constexpr duration` (const `_Rep2` &\_\_rep)
- `template<typename _Rep2 , typename _Period2 , typename = typename enable_if<treat_as_floating_point<rep>::value || (ratio_divide<_Period2, period>::type::den == 1 && !treat_as_floating_point<_Rep2>::value)>::type> constexpr duration` (const [duration](#)< `_Rep2`, `_Period2` > &\_\_d)
- `constexpr rep count` () const
- `template<typename _Rep2 = rep> enable\_if<!treat\_as\_floating\_point<\_Rep2>::value, duration & >::type operator%=` (const `rep` &\_\_rhs)
- `template<typename _Rep2 = rep> enable\_if<!treat\_as\_floating\_point<\_Rep2>::value, duration & >::type operator%=` (const [duration](#) &\_\_d)

- `duration & operator*=` (const rep &\_\_rhs)
- constexpr `duration operator+` () const
- `duration & operator++` ()
- `duration operator++` (int)
- `duration & operator+=` (const `duration` &\_\_d)
- constexpr `duration operator-` () const
- `duration & operator--` ()
- `duration operator--` (int)
- `duration & operator-=` (const `duration` &\_\_d)
- `duration & operator/=` (const rep &\_\_rhs)
- `duration & operator=` (const `duration` &)
- `static_assert` (\_Period::num > 0, "period must be positive")
- `static_assert` (!\_\_is\_duration<\_Rep>::value, "rep cannot be a `duration`")
- `static_assert` (\_\_is\_ratio<\_Period>::value, "period must be a specialization of `ratio`")

#### Static Public Member Functions

- static constexpr `duration max` ()
- static constexpr `duration min` ()
- static constexpr `duration zero` ()

##### 5.412.1 Detailed Description

`template<typename _Rep, typename _Period> struct std::chrono::duration<_Rep, _Period>`

`duration`

Definition at line 221 of file `chrono`.

The documentation for this struct was generated from the following file:

- `chrono`

#### 5.413 `std::chrono::duration_values<_Rep>` Struct Template Reference

`duration_values`

### Static Public Member Functions

- static constexpr `_Rep` **max** ()
- static constexpr `_Rep` **min** ()
- static constexpr `_Rep` **zero** ()

#### 5.413.1 Detailed Description

`template<typename _Rep> struct std::chrono::duration_values< _Rep >`

[duration\\_values](#)

Definition at line 194 of file `chrono`.

The documentation for this struct was generated from the following file:

- [chrono](#)

## 5.414 `std::chrono::system_clock` Struct Reference

[system\\_clock](#)

### Public Types

- typedef [chrono::seconds](#) **duration**
- typedef `duration::period` **period**
- typedef `duration::rep` **rep**
- typedef [chrono::time\\_point](#)< [system\\_clock](#), [duration](#) > **time\_point**

### Public Member Functions

- **static\_assert** (`system_clock::duration::min()`< `system_clock::duration::zero()`, "a clock's minimum [duration](#) cannot be [less](#) than its epoch")

### Static Public Member Functions

- static [time\\_point](#) **from\_time\_t** (`std::time_t __t`)
- static [time\\_point](#) **now** () throw ()
- static `std::time_t` **to\_time\_t** (`const time\_point &__t`)

## 5.415 `std::chrono::time_point< _Clock, _Dur >` Struct Template Reference 2411

### Static Public Attributes

- static constexpr bool `is_monotonic`

#### 5.414.1 Detailed Description

[system\\_clock](#)

Definition at line 653 of file `chrono`.

The documentation for this struct was generated from the following file:

- [chrono](#)

## 5.415 `std::chrono::time_point< _Clock, _Dur >` Struct Template Reference

[time\\_point](#)

### Public Types

- typedef `_Clock` **clock**
- typedef `_Dur` **duration**
- typedef `duration::period` **period**
- typedef `duration::rep` **rep**

### Public Member Functions

- constexpr **time\_point** (const duration &\_\_dur)
- template<typename `_Dur2` >  
constexpr **time\_point** (const [time\\_point](#)< clock, `_Dur2` > &\_\_t)
- [time\\_point](#) & **operator+=** (const duration &\_\_dur)
- [time\\_point](#) & **operator-=** (const duration &\_\_dur)
- constexpr duration **time\_since\_epoch** () const

### Static Public Member Functions

- static constexpr [time\\_point](#) **max** ()
- static constexpr [time\\_point](#) **min** ()

## 2412

```
template<typename _Clock, typename _Dur> struct std::chrono::time_point<
_Clock, _Dur >
```

Definition at line 518 of file chrono.

- chrono

treat\_as\_floating\_point

[illegible]

- typedef **integral\_constant**< bool, \_\_v > **type**
- typedef bool **value\_type**

- **constexpr operator value\_type ()**

- static constexpr bool **value**

```
template<typename _Rep> struct std::chrono::treat_as_floating_point< _Rep >
```

Definition at line 188 of file chrono.

### 5.417 `std::codecvt< _InternT, _ExternT, _StateT >` Class Template Reference

The documentation for this struct was generated from the following file:

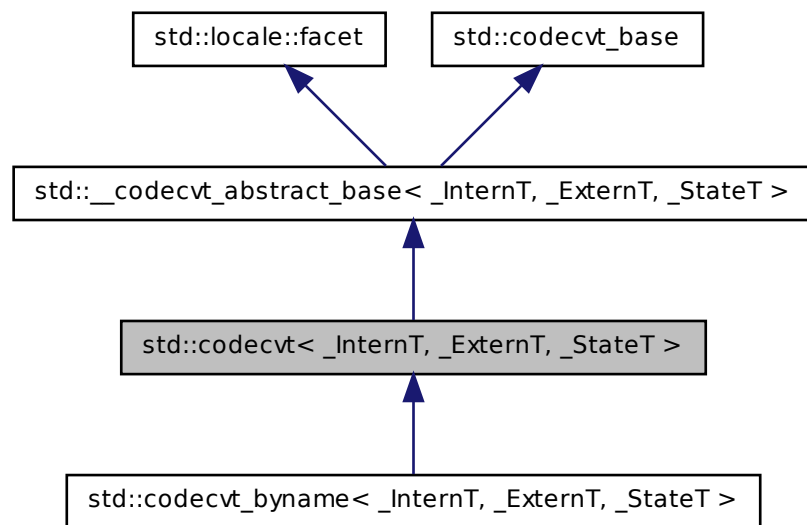
- [chrono](#)

### 5.417 `std::codecvt< _InternT, _ExternT, _StateT >` Class Template Reference

Primary class template `codecvt`.

NB: Generic, mostly useless implementation.

Inheritance diagram for `std::codecvt< _InternT, _ExternT, _StateT >`:



#### Public Types

- typedef `_ExternT` **extern\_type**
- typedef `_InternT` **intern\_type**
- typedef `codecvt_base::result` **result**
- typedef `_StateT` **state\_type**



### Public Member Functions

- **codecvt** (size\_t \_\_refs=0)
- **codecvt** (\_\_c\_locale \_\_cloc, size\_t \_\_refs=0)
- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*&\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const

### Static Public Attributes

- static `locale::id` **id**

### Protected Member Functions

- virtual bool **do\_always\_noconv** () const throw ()
- virtual int **do\_encoding** () const throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*&\_\_to\_next) const
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int **do\_max\_length** () const throw ()
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const

### Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)

## 5.417 `std::codecvt<_InternT, _ExternT, _StateT >` Class Template Reference

- static void `_S_destroy_c_locale` (`__c_locale &__cloc`)
- static `__c_locale` `_S_get_c_locale` ()
- static const char \* `_S_get_c_name` () throw ()
- static `__c_locale` `_S_lc_ctype_c_locale` (`__c_locale __cloc`, const char \* `__s`)

### Protected Attributes

- `__c_locale` `_M_c_locale_codecvt`

### Friends

- class `locale::Impl`

#### 5.417.1 Detailed Description

`template<typename _InternT, typename _ExternT, typename _StateT> class std::codecvt<_InternT, _ExternT, _StateT >`

Primary class template `codecvt`.

NB: Generic, mostly useless implementation.

Definition at line 277 of file `codecvt.h`.

#### 5.417.2 Member Function Documentation

5.417.2.1 `template<typename _InternT, typename _ExternT, typename _StateT> virtual result std::codecvt<_InternT, _ExternT, _StateT>::do_out ( state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type *& __from_next, extern_type * __to, extern_type * __to_end, extern_type *& __to_next ) const` [`protected`, `virtual`]

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

### See also

[out](#) for more information.

Implements `std::__codecvt_abstract_base<_InternT, _ExternT, _StateT >`.

**5.417.2.2** `template<typename _InternT, typename _ExternT, typename  
_StateT> result std::__codecvt_abstract_base< _InternT, _ExternT,  
_StateT >::in ( state_type & __state, const extern_type * __from,  
const extern_type * __from_end, const extern_type *& __from_next,  
intern_type * __to, intern_type * __to_end, intern_type *&  
__to_next ) const [inline, inherited]`

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

#### Parameters

*state* Persistent conversion state data.  
*from* Start of input.  
*from\_end* End of input.  
*from\_next* Returns start of unconverted data.  
*to* Start of output buffer.  
*to\_end* End of output buffer.  
*to\_next* Returns start of unused output area.

#### Returns

`codecvt_base::result`.

Definition at line 197 of file `codecvt.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::underflow()`.

**5.417.2.3** `template<typename _InternT, typename _ExternT, typename  
_StateT> result std::__codecvt_abstract_base<_InternT, _ExternT,  
_StateT >::out ( state_type & __state, const intern_type * __from,  
const intern_type * __from_end, const intern_type *& __from_next,  
extern_type * __to, extern_type * __to_end, extern_type *&  
__to_next ) const [inline, inherited]`

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

### Parameters

- state* Persistent conversion state data.
- from* Start of input.
- from\_end* End of input.
- from\_next* Returns start of unconverted data.
- to* Start of output buffer.
- to\_end* End of output buffer.
- to\_next* Returns start of unused output area.

### Returns

`codecvt_base::result`.

Definition at line 117 of file `codecvt.h`.

5.417.2.4 `template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::unshift ( state_type & __state, extern_type * __to, extern_type * __to_end, extern_type *& __to_next ) const`  
**[inline, inherited]**

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

#### Parameters

- state* Persistent conversion state data.
- to* Start of output buffer.
- to\_end* End of output buffer.
- to\_next* Returns start of unused output area.

#### Returns

`codecvt_base::result`.

Definition at line 156 of file `codecvt.h`.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

## 5.418 `std::codecvt< _InternT, _ExternT, encoding_state >` Class Template Reference

`codecvt<InternT, _ExternT, encoding_state>` specialization.

Inheritance diagram for `std::codecvt< _InternT, _ExternT, encoding_state >`:



## Public Types

- typedef `state_type::descriptor_type` **descriptor\_type**
- typedef `_ExternT` **extern\_type**
- typedef `_InternT` **intern\_type**
- typedef `codecvt_base::result` **result**
- typedef `__gnu_cxx::encoding_state` **state\_type**

## Public Member Functions

- **codecvt** (`size_t __refs=0`)
- **codecvt** (`state_type &__enc, size_t __refs=0`)
- **bool always\_noconv** () const throw ()
- **int encoding** () const throw ()
- **result in** (`state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next`) const
- **int length** (`state_type &__state, const extern_type *__from, const extern_type *__from_end, size_t __max`) const
- **int max\_length** () const throw ()
- **result out** (`state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next`) const
- **result unshift** (`state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next`) const

## Static Public Attributes

- static `locale::id` **id**

## Protected Member Functions

- virtual **bool do\_always\_noconv** () const throw ()
- virtual **int do\_encoding** () const throw ()

- virtual result `do_in` (`state_type` &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*&\_\_to\_next) const
- virtual int `do_length` (`state_type` &, const extern\_type \*\_\_from, const extern\_type \*\_\_to\_end, size\_t \_\_max) const
- virtual int `do_max_length` () const throw ()
- virtual result `do_out` (`state_type` &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const
- virtual result `do_unshift` (`state_type` &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const

#### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale` (\_\_c\_locale &\_\_cloc) throw ()
- static void `_S_create_c_locale` (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void `_S_destroy_c_locale` (\_\_c\_locale &\_\_cloc)
- static `__c_locale _S_get_c_locale` ()
- static const char \* `_S_get_c_name` () throw ()
- static `__c_locale _S_lc_ctype_c_locale` (\_\_c\_locale \_\_cloc, const char \*\_\_s)

#### Friends

- class `locale::Impl`

#### 5.418.1 Detailed Description

`template<typename _InternT, typename _ExternT> class std::codecvt< _InternT, _ExternT, encoding_state >`

`codecvt<InternT, _ExternT, encoding_state>` specialization.

Definition at line 232 of file `codecvt_specializations.h`.

## 5.418.2 Member Function Documentation

**5.418.2.1** `template<typename _InternT, typename _ExternT >  
codecvt_base::result std::codecvt<_InternT, _ExternT,  
encoding_state>::do_out ( state_type & __state, const intern_type  
* __from, const intern_type * __from_end, const intern_type  
*& __from_next, extern_type * __to, extern_type * __to_end,  
extern_type *& __to_next ) const [protected, virtual]`

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

### See also

[out](#) for more information.

Implements `std::__codecvt_abstract_base<_InternT, _ExternT, encoding_state>`.

Definition at line 308 of file `codecvt_specializations.h`.

**5.418.2.2** `result std::__codecvt_abstract_base<_InternT, _ExternT,  
encoding_state>::in ( state_type & __state, const extern_type  
* __from, const extern_type * __from_end, const extern_type  
*& __from_next, intern_type * __to, intern_type * __to_end,  
intern_type *& __to_next ) const [inline, inherited]`

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`.



If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

#### Parameters

*state* Persistent conversion state data.  
*from* Start of input.  
*from\_end* End of input.  
*from\_next* Returns start of unconverted data.  
*to* Start of output buffer.  
*to\_end* End of output buffer.  
*to\_next* Returns start of unused output area.

#### Returns

`codecvt_base::result`.

Definition at line 197 of file `codecvt.h`.

**5.418.2.3** `result std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >::out ( state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type * __to, const intern_type * __to_end, const intern_type * __to_next ) const` `[inline, inherited]`

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

### Parameters

*state* Persistent conversion state data.  
*from* Start of input.  
*from\_end* End of input.  
*from\_next* Returns start of unconverted data.  
*to* Start of output buffer.  
*to\_end* End of output buffer.  
*to\_next* Returns start of unused output area.

### Returns

codecvt\_base::result.

Definition at line 117 of file codecvt.h.

References std::\_\_codecvt\_abstract\_base< \_InternT, \_ExternT, \_StateT >::do\_out().

**5.418.2.4 result std::\_\_codecvt\_abstract\_base< \_InternT, \_ExternT, encoding\_state >::unshift ( state\_type & \_\_state, extern\_type \* \_\_to, extern\_type \* \_\_to\_end, extern\_type \*& \_\_to\_next ) const [inline, inherited]**

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do\_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of codecvt\_base::result. If the state could be reset and data written, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the output has insufficient space, returns codecvt\_base::partial. Otherwise the reset failed and codecvt\_base::error is returned.

### Parameters

*state* Persistent conversion state data.  
*to* Start of output buffer.

*to\_end* End of output buffer.

*to\_next* Returns start of unused output area.

### Returns

`codecvt_base::result`.

Definition at line 156 of file `codecvt.h`.

The documentation for this class was generated from the following file:

- [codecvt\\_specializations.h](#)

## 5.419 `std::codecvt< char, char, mbstate_t >` Class Template Reference

class `codecvt<char, char, mbstate_t>` specialization.

Inheritance diagram for `std::codecvt< char, char, mbstate_t >`:



### Public Types

- typedef char **extern\_type**
- typedef char **intern\_type**
- typedef `codecvt_base::result` **result**
- typedef `mbstate_t` **state\_type**

### Public Member Functions

- **codecvt** (`size_t __refs=0`)
- **codecvt** (`__c_locale __cloc, size_t __refs=0`)
- **bool always\_noconv** () const throw ()
- **int encoding** () const throw ()
- **result in** (`state_type &__state, const extern_type * __from, const extern_type * __from_end, const extern_type *& __from_next, intern_type * __to, intern_type * __to_end, intern_type *& __to_next`) const
- **int length** (`state_type &__state, const extern_type * __from, const extern_type * __end, size_t __max`) const

- `int max_length () const throw ()`
- result `out` (`state_type &__state`, `const intern_type *__from`, `const intern_type *__from_end`, `const intern_type *&__from_next`, `extern_type *__to`, `extern_type *__to_end`, `extern_type *&__to_next`) `const`
- result `unshift` (`state_type &__state`, `extern_type *__to`, `extern_type *__to_end`, `extern_type *&__to_next`) `const`

### Static Public Attributes

- static `locale::id id`

### Protected Member Functions

- virtual `bool do_always_noconv () const throw ()`
- virtual `int do_encoding () const throw ()`
- virtual result `do_in` (`state_type &__state`, `const extern_type *__from`, `const extern_type *__from_end`, `const extern_type *&__from_next`, `intern_type *__to`, `intern_type *__to_end`, `intern_type *&__to_next`) `const`
- virtual `int do_length` (`state_type &`, `const extern_type *__from`, `const extern_type *__end`, `size_t __max`) `const`
- virtual `int do_max_length () const throw ()`
- virtual result `do_out` (`state_type &__state`, `const intern_type *__from`, `const intern_type *__from_end`, `const intern_type *&__from_next`, `extern_type *__to`, `extern_type *__to_end`, `extern_type *&__to_next`) `const`
- virtual result `do_unshift` (`state_type &__state`, `extern_type *__to`, `extern_type *__to_end`, `extern_type *&__to_next`) `const`

### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static `void _S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static `void _S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static `const char * _S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

### Protected Attributes

- `__c_locale _M_c_locale_codecvt`

## Friends

- class `locale::_Impl`

### 5.419.1 Detailed Description

`template<> class std::codecvt< char, char, mbstate_t >`

class `codecvt<char, char, mbstate_t>` specialization.

Definition at line 339 of file `codecvt.h`.

### 5.419.2 Member Function Documentation

**5.419.2.1** `virtual result std::codecvt< char, char, mbstate_t >::do_out ( state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type *& __from_next, extern_type * __to, extern_type * __to_end, extern_type *& __to_next ) const [protected, virtual]`

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

#### See also

[out](#) for more information.

Implements `std::__codecvt_abstract_base< char, char, mbstate_t >`.

**5.419.2.2** `result std::__codecvt_abstract_base< char, char, mbstate_t >::in ( state_type & __state, const extern_type * __from, const extern_type * __from_end, const extern_type *& __from_next, intern_type * __to, intern_type * __to_end, intern_type *& __to_next ) const [inline, inherited]`

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

### Parameters

*state* Persistent conversion state data.  
*from* Start of input.  
*from\_end* End of input.  
*from\_next* Returns start of unconverted data.  
*to* Start of output buffer.  
*to\_end* End of output buffer.  
*to\_next* Returns start of unused output area.

### Returns

`codecvt_base::result`.

Definition at line 197 of file `codecvt.h`.

**5.419.2.3** `result std::__codecvt_abstract_base< char , char , mbstate_t >::out ( state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type *& __from_next, extern_type * __to, extern_type * __to_end, extern_type *& __to_next ) const [inline, inherited]`

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted

character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

#### Parameters

*state* Persistent conversion state data.  
*from* Start of input.  
*from\_end* End of input.  
*from\_next* Returns start of unconverted data.  
*to* Start of output buffer.  
*to\_end* End of output buffer.  
*to\_next* Returns start of unused output area.

#### Returns

`codecvt_base::result`.

Definition at line 117 of file `codecvt.h`.

**5.419.2.4** `result std::__codecvt_abstract_base< char , char , mbstate_t  
>::unshift ( state_type & __state, extern_type * __to, extern_type  
* __to_end, extern_type *& __to_next ) const [inline,  
inherited]`

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

## 5.420 `std::codecvt< wchar_t, char, mbstate_t >` Class Template Reference 2429

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

### Parameters

- state* Persistent conversion state data.
- to* Start of output buffer.
- to\_end* End of output buffer.
- to\_next* Returns start of unused output area.

### Returns

`codecvt_base::result`.

Definition at line 156 of file `codecvt.h`.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

## 5.420 `std::codecvt< wchar_t, char, mbstate_t >` Class Template Reference

class `codecvt<wchar_t, char, mbstate_t>` specialization.

Inheritance diagram for `std::codecvt< wchar_t, char, mbstate_t >`:



### Public Types

- typedef char **extern\_type**
- typedef wchar\_t **intern\_type**
- typedef `codecvt_base::result` **result**
- typedef `mbstate_t` **state\_type**



### Public Member Functions

- **codecvt** (size\_t \_\_refs=0)
- **codecvt** (\_\_c\_locale \_\_cloc, size\_t \_\_refs=0)
- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*&\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const

### Static Public Attributes

- static `locale::id` **id**

### Protected Member Functions

- virtual bool **do\_always\_noconv** () const throw ()
- virtual int **do\_encoding** () const throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*&\_\_to\_next) const
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int **do\_max\_length** () const throw ()
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const

### Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)

## 5.420 `std::codecvt< wchar_t, char, mbstate_t >` Class Template Reference 2431

---

- static void `_S_destroy_c_locale` (`__c_locale` & `__cloc`)
- static `__c_locale` `_S_get_c_locale` ()
- static const char \* `_S_get_c_name` () throw ()
- static `__c_locale` `_S_lc_ctype_c_locale` (`__c_locale` `__cloc`, const char \* `__s`)

### Protected Attributes

- `__c_locale` `_M_c_locale_codecvt`

### Friends

- class `locale::_Impl`

### 5.420.1 Detailed Description

`template<> class std::codecvt< wchar_t, char, mbstate_t >`

class `codecvt<wchar_t, char, mbstate_t>` specialization.

Definition at line 397 of file `codecvt.h`.

### 5.420.2 Member Function Documentation

**5.420.2.1** virtual result `std::codecvt< wchar_t, char, mbstate_t >::do_out ( state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type *& __from_next, extern_type * __to, extern_type * __to_end, extern_type *& __to_next ) const` **[protected, virtual]**

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

### See also

[out](#) for more information.

Implements `std::__codecvt_abstract_base< wchar_t, char, mbstate_t >`.

**5.420.2.2** `result std::__codecvt_abstract_base< wchar_t , char , mbstate_t >::in ( state_type & __state, const extern_type * __from, const extern_type * __from_end, const extern_type *& __from_next, intern_type * __to, intern_type * __to_end, intern_type *& __to_next ) const` [**inline, inherited**]

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

#### Parameters

- state* Persistent conversion state data.
- from* Start of input.
- from\_end* End of input.
- from\_next* Returns start of unconverted data.
- to* Start of output buffer.
- to\_end* End of output buffer.
- to\_next* Returns start of unused output area.

#### Returns

`codecvt_base::result`.

Definition at line 197 of file `codecvt.h`.

**5.420.2.3** `result std::__codecvt_abstract_base< wchar_t , char , mbstate_t  
>::out ( state_type & __state, const intern_type * __from, const  
intern_type * __from_end, const intern_type *& __from_next,  
extern_type * __to, extern_type * __to_end, extern_type *&  
__to_next ) const [inline, inherited]`

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

#### Parameters

*state* Persistent conversion state data.

*from* Start of input.

*from\_end* End of input.

*from\_next* Returns start of unconverted data.

*to* Start of output buffer.

*to\_end* End of output buffer.

*to\_next* Returns start of unused output area.

#### Returns

`codecvt_base::result`.

Definition at line 117 of file `codecvt.h`.

**5.420.2.4** result std::\_\_codecvt\_abstract\_base< wchar\_t , char , mbstate\_t  
 >::unshift ( state\_type & \_\_state, extern\_type \* \_\_to, extern\_type  
 \* \_\_to\_end, extern\_type \*& \_\_to\_next ) const [inline,  
 inherited]

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do\_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of codecvt\_base::result. If the state could be reset and data written, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the output has insufficient space, returns codecvt\_base::partial. Otherwise the reset failed and codecvt\_base::error is returned.

### Parameters

- state* Persistent conversion state data.
- to* Start of output buffer.
- to\_end* End of output buffer.
- to\_next* Returns start of unused output area.

### Returns

codecvt\_base::result.

Definition at line 156 of file codecvt.h.

The documentation for this class was generated from the following file:

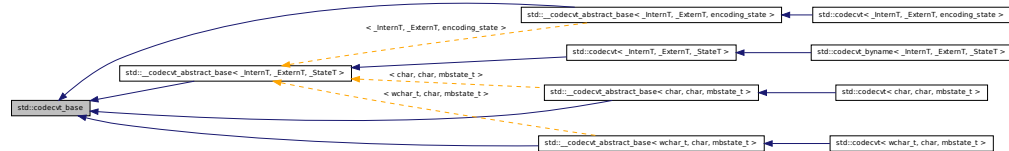
- [codecvt.h](#)

## 5.421 std::codecvt\_base Class Reference

Empty base class for codecvt facet [22.2.1.5].

## 5.422 `std::codecvt_byname<_InternT, _ExternT, _StateT>` Class Template Reference 2435

Inheritance diagram for `std::codecvt_base`:



### Public Types

- enum `result` { `ok`, `partial`, `error`, `noconv` }

#### 5.421.1 Detailed Description

Empty base class for `codecvt` facet [22.2.1.5].

Definition at line 47 of file `codecvt.h`.

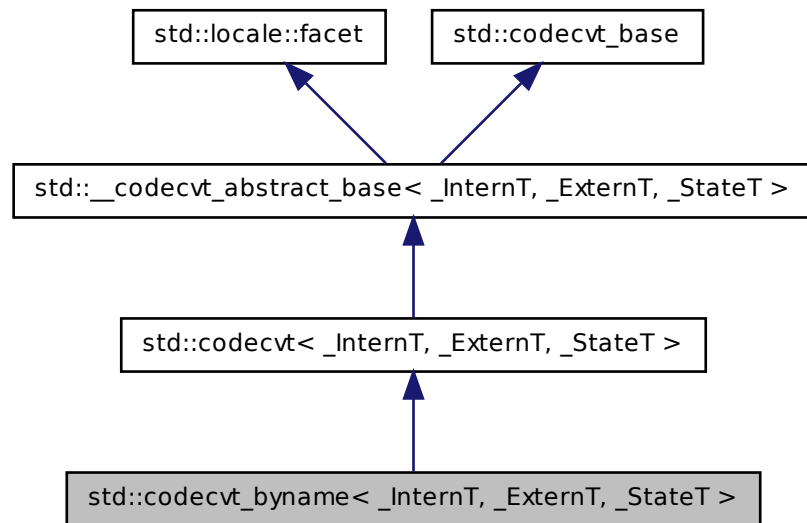
The documentation for this class was generated from the following file:

- [codecvt.h](#)

## 5.422 `std::codecvt_byname<_InternT, _ExternT, _StateT>` Class Template Reference

class `codecvt_byname` [22.2.1.6].

Inheritance diagram for `std::codecvt_byname< _InternT, _ExternT, _StateT >`:



### Public Types

- typedef `_ExternT` **extern\_type**
- typedef `_InternT` **intern\_type**
- typedef `codecvt_base::result` **result**
- typedef `_StateT` **state\_type**

### Public Member Functions

- **codecvt\_byname** (`const char *__s`, `size_t __refs=0`)
- **bool always\_noconv** () `const throw ()`
- **int encoding** () `const throw ()`
- **result in** (`state_type &__state`, `const extern_type *__from`, `const extern_type *__from_end`, `const extern_type *&__from_next`, `intern_type *__to`, `intern_type *__to_end`, `intern_type *&__to_next`) `const`
- **int length** (`state_type &__state`, `const extern_type *__from`, `const extern_type *__end`, `size_t __max`) `const`

- `int max_length () const throw ()`
- result `out` (`state_type &__state`, `const intern_type *__from`, `const intern_type *__from_end`, `const intern_type *&__from_next`, `extern_type *__to`, `extern_type *__to_end`, `extern_type *&__to_next`) `const`
- result `unshift` (`state_type &__state`, `extern_type *__to`, `extern_type *__to_end`, `extern_type *&__to_next`) `const`

#### Static Public Attributes

- static `locale::id id`

#### Protected Member Functions

- virtual `bool do_always_noconv () const throw ()`
- virtual `int do_encoding () const throw ()`
- virtual result `do_in` (`state_type &__state`, `const extern_type *__from`, `const extern_type *__from_end`, `const extern_type *&__from_next`, `intern_type *__to`, `intern_type *__to_end`, `intern_type *&__to_next`) `const`
- virtual `int do_length` (`state_type &`, `const extern_type *__from`, `const extern_type *__end`, `size_t __max`) `const`
- virtual `int do_max_length () const throw ()`
- virtual result `do_out` (`state_type &__state`, `const intern_type *__from`, `const intern_type *__from_end`, `const intern_type *&__from_next`, `extern_type *__to`, `extern_type *__to_end`, `extern_type *&__to_next`) `const`
- virtual result `do_unshift` (`state_type &__state`, `extern_type *__to`, `extern_type *__to_end`, `extern_type *&__to_next`) `const`

#### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static `void _S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static `void _S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static `const char * _S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

#### Protected Attributes

- `__c_locale _M_c_locale_codecvt`



## Friends

- class `locale::_Impl`

### 5.422.1 Detailed Description

`template<typename _InternT, typename _ExternT, typename _StateT> class std::codecvt_byname<_InternT, _ExternT, _StateT>`

class `codecvt_byname` [22.2.1.6].

Definition at line 457 of file `codecvt.h`.

### 5.422.2 Member Function Documentation

**5.422.2.1** `template<typename _InternT, typename _ExternT, typename _StateT> virtual result std::codecvt<_InternT, _ExternT, _StateT>::do_out ( state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type *& __from_next, extern_type * __to, extern_type * __to_end, extern_type *& __to_next ) const` [`protected`, `virtual`, `inherited`]

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

#### See also

[out](#) for more information.

Implements `std::__codecvt_abstract_base<_InternT, _ExternT, _StateT>`.

**5.422.2.2** `template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base<_InternT, _ExternT, _StateT>::in ( state_type & __state, const extern_type * __from, const extern_type * __from_end, const extern_type *& __from_next, intern_type * __to, intern_type * __to_end, intern_type *& __to_next ) const` [`inline`, `inherited`]

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

#### Parameters

*state* Persistent conversion state data.  
*from* Start of input.  
*from\_end* End of input.  
*from\_next* Returns start of unconverted data.  
*to* Start of output buffer.  
*to\_end* End of output buffer.  
*to\_next* Returns start of unused output area.

#### Returns

`codecvt_base::result`.

Definition at line 197 of file `codecvt.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::underflow()`.

**5.422.2.3** `template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::out ( state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type *& __from_next, extern_type * __to, extern_type * __to_end, extern_type *& __to_next ) const` [`inline, inherited`]

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

#### Parameters

*state* Persistent conversion state data.  
*from* Start of input.  
*from\_end* End of input.  
*from\_next* Returns start of unconverted data.  
*to* Start of output buffer.  
*to\_end* End of output buffer.  
*to\_next* Returns start of unused output area.

#### Returns

`codecvt_base::result`.

Definition at line 117 of file `codecvt.h`.

```
5.422.2.4 template<typename _InternT, typename _ExternT, typename
 _StateT> result std::__codecvt_abstract_base<_InternT, _ExternT,
 _StateT>::unshift (state_type & __state, extern_type * __to,
 extern_type * __to_end, extern_type *& __to_next) const
 [inline, inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

#### Parameters

- state* Persistent conversion state data.
- to* Start of output buffer.
- to\_end* End of output buffer.
- to\_next* Returns start of unused output area.

#### Returns

`codecvt_base::result`.

Definition at line 156 of file `codecvt.h`.

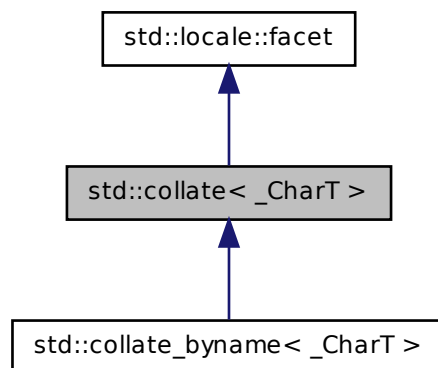
The documentation for this class was generated from the following file:

- [codecvt.h](#)

### 5.423 `std::collate<_CharT>` Class Template Reference

Facet for localized string comparison.

Inheritance diagram for `std::collate<_CharT>`:



### Public Types

- typedef `_CharT` [char\\_type](#)
- typedef `basic_string<_CharT>` [string\\_type](#)

### Public Member Functions

- [collate](#) (`size_t __refs=0`)
- [collate](#) (`__c_locale __cloc, size_t __refs=0`)
- `template<>`  
`int _M_compare (const char *, const char *) const throw()`
- `template<>`  
`int _M_compare (const wchar_t *, const wchar_t *) const throw()`
- `int _M_compare (const _CharT *, const _CharT *) const throw ()`
- `template<>`  
`size_t _M_transform (char *, const char *, size_t) const throw()`
- `template<>`  
`size_t _M_transform (wchar_t *, const wchar_t *, size_t) const throw()`
- `size_t _M_transform (_CharT *, const _CharT *, size_t) const throw ()`
- `int compare (const _CharT *__lo1, const _CharT *__hi1, const _CharT *__lo2, const _CharT *__hi2) const`
- `long hash (const _CharT *__lo, const _CharT *__hi) const`
- [string\\_type transform](#) (`const _CharT *__lo, const _CharT *__hi`) const

**Static Public Attributes**

- static [locale::id](#) id

**Protected Member Functions**

- virtual [~collate](#) ()
- virtual int [do\\_compare](#) (const \_CharT \*\_\_lo1, const \_CharT \*\_\_hi1, const \_CharT \*\_\_lo2, const \_CharT \*\_\_hi2) const
- virtual long [do\\_hash](#) (const \_CharT \*\_\_lo, const \_CharT \*\_\_hi) const
- virtual [string\\_type](#) [do\\_transform](#) (const \_CharT \*\_\_lo, const \_CharT \*\_\_hi) const

**Static Protected Member Functions**

- static \_\_c\_locale [\\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale [\\_S\\_get\\_c\\_locale](#) ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static \_\_c\_locale [\\_S\\_lc\\_ctype\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \*\_\_s)

**Protected Attributes**

- \_\_c\_locale [\\_M\\_c\\_locale\\_collate](#)

**Friends**

- class [locale::Impl](#)

**5.423.1 Detailed Description**

**template<typename \_CharT> class std::collate<\_CharT>**

Facet for localized string comparison. This facet encapsulates the code to compare strings in a localized manner.

The collate template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the collate facet.

Definition at line 618 of file locale\_classes.h.

### 5.423.2 Member Typedef Documentation

#### 5.423.2.1 `template<typename _CharT> typedef _CharT std::collate<_CharT>::char_type`

Public typedefs.

Reimplemented in [std::collate\\_byname<\\_CharT>](#).

Definition at line 624 of file locale\_classes.h.

#### 5.423.2.2 `template<typename _CharT> typedef basic_string<_CharT> std::collate<_CharT>::string_type`

Public typedefs.

Reimplemented in [std::collate\\_byname<\\_CharT>](#).

Definition at line 625 of file locale\_classes.h.

### 5.423.3 Constructor & Destructor Documentation

#### 5.423.3.1 `template<typename _CharT> std::collate<_CharT>::collate ( size_t __refs = 0 ) [inline, explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

##### Parameters

*refs* Passed to the base facet class.

Definition at line 645 of file locale\_classes.h.

#### 5.423.3.2 `template<typename _CharT> std::collate<_CharT>::collate ( _c_locale __cloc, size_t __refs = 0 ) [inline, explicit]`

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

**Parameters**

*cloc* The C locale.

*refs* Passed to the base facet class.

Definition at line 659 of file locale\_classes.h.

**5.423.3.3** `template<typename _CharT> virtual std::collate<_CharT>::~~collate( ) [inline, protected, virtual]`

Destructor.

Definition at line 722 of file locale\_classes.h.

**5.423.4 Member Function Documentation**

**5.423.4.1** `template<typename _CharT> int std::collate<_CharT>::compare( const _CharT * __lo1, const _CharT * __hi1, const _CharT * __lo2, const _CharT * __hi2 ) const [inline]`

Compare two strings.

This function compares two strings and returns the result by calling [collate::do\\_compare\(\)](#).

**Parameters**

*lo1* Start of string 1.

*hi1* End of string 1.

*lo2* Start of string 2.

*hi2* End of string 2.

**Returns**

1 if string1 > string2, -1 if string1 < string2, else 0.

Definition at line 676 of file locale\_classes.h.



**5.423.4.2** `template<typename _CharT> int std::collate<_CharT  
>::do_compare ( const _CharT * __lo1, const _CharT *  
__hi1, const _CharT * __lo2, const _CharT * __hi2 ) const  
[protected, virtual]`

Compare two strings.

This function is a hook for derived classes to change the value returned.

**See also**

[compare\(\)](#).

#### Parameters

*lo1* Start of string 1.

*hi1* End of string 1.

*lo2* Start of string 2.

*hi2* End of string 2.

#### Returns

1 if string1 > string2, -1 if string1 < string2, else 0.

Definition at line 136 of file locale\_classes.tcc.

References `std::basic_string<_CharT, _Traits, _Alloc >::c_str()`, `std::basic_string<_CharT, _Traits, _Alloc >::data()`, and `std::basic_string<_CharT, _Traits, _Alloc >::length()`.

**5.423.4.3** `template<typename _CharT> long std::collate<_CharT  
>::do_hash ( const _CharT * __lo, const _CharT * __hi ) const  
[protected, virtual]`

Return hash of a string.

This function computes and returns a hash on the input string. This function is a hook for derived classes to change the value returned.

#### Parameters

*lo* Start of string.

*hi* End of string.

**Returns**

Hash value.

Definition at line 231 of file locale\_classes.tcc.

**5.423.4.4** `template<typename _CharT> collate<_CharT>::string_type  
std::collate<_CharT>::do_transform ( const _CharT * __lo, const  
_CharT * __hi ) const [protected, virtual]`

Transform string to comparable form.

This function is a hook for derived classes to change the value returned.

**Parameters**

*lo1* Start of string 1.

*hi1* End of string 1.

*lo2* Start of string 2.

*hi2* End of string 2.

**Returns**

1 if string1 > string2, -1 if string1 < string2, else 0.

Definition at line 175 of file locale\_classes.tcc.

References `std::basic_string<_CharT, _Traits, _Alloc>::append()`, `std::basic_string<_CharT, _Traits, _Alloc>::c_str()`, `std::basic_string<_CharT, _Traits, _Alloc>::data()`, `std::basic_string<_CharT, _Traits, _Alloc>::length()`, and `std::basic_string<_CharT, _Traits, _Alloc>::push_back()`.

**5.423.4.5** `template<typename _CharT> long std::collate<_CharT>::hash (  
const _CharT * __lo, const _CharT * __hi ) const [inline]`

Return hash of a string.

This function computes and returns a hash on the input string. It does so by returning [collate::do\\_hash\(\)](#).

**Parameters**

*lo* Start of string.

*hi* End of string.

### Returns

Hash value.

Definition at line 709 of file `locale_classes.h`.

**5.423.4.6** `template<typename _CharT> string_type std::collate<_CharT  
>::transform ( const _CharT * __lo, const _CharT * __hi ) const  
[inline]`

Transform string to comparable form.

This function is a wrapper for `strxfrm` functionality. It takes the input string and returns a modified string that can be directly compared to other transformed strings. In the C locale, this function just returns a copy of the input string. In some other locales, it may replace two chars with one, change a char for another, etc. It does so by returning [collate::do\\_transform\(\)](#).

### Parameters

*lo* Start of string.

*hi* End of string.

### Returns

Transformed `string_type`.

Definition at line 695 of file `locale_classes.h`.

Referenced by `std::regex_traits<_Ch_type>::transform()`.

## 5.423.5 Member Data Documentation

**5.423.5.1** `template<typename _CharT> locale::id std::collate<_CharT>::id  
[static]`

Numpunct facet id.

Definition at line 635 of file `locale_classes.h`.

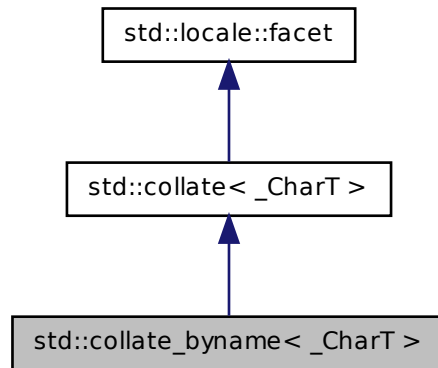
The documentation for this class was generated from the following files:

- [locale\\_classes.h](#)
- [locale\\_classes.tcc](#)

**5.424 std::collate\_byname< \_CharT > Class Template Reference**

class [collate\\_byname](#) [22.2.4.2].

Inheritance diagram for std::collate\_byname< \_CharT >:

**Public Types**

- typedef `_CharT` [char\\_type](#)
- typedef [basic\\_string](#)< `_CharT` > [string\\_type](#)

**Public Member Functions**

- **collate\_byname** (const char \* \_\_s, size\_t \_\_refs=0)
- int **\_M\_compare** (const \_CharT \*, const \_CharT \*) const throw ()
- template<>  
int **\_M\_compare** (const char \*, const char \*) const throw()
- template<>  
int **\_M\_compare** (const wchar\_t \*, const wchar\_t \*) const throw()
- size\_t **\_M\_transform** (\_CharT \*, const \_CharT \*, size\_t) const throw ()
- template<>  
size\_t **\_M\_transform** (char \*, const char \*, size\_t) const throw()
- template<>  
size\_t **\_M\_transform** (wchar\_t \*, const wchar\_t \*, size\_t) const throw()

- int [compare](#) (const \_CharT \*\_\_lo1, const \_CharT \*\_\_hi1, const \_CharT \*\_\_lo2, const \_CharT \*\_\_hi2) const
- long [hash](#) (const \_CharT \*\_\_lo, const \_CharT \*\_\_hi) const
- [string\\_type transform](#) (const \_CharT \*\_\_lo, const \_CharT \*\_\_hi) const

### Static Public Attributes

- static [locale::id id](#)

### Protected Member Functions

- virtual int [do\\_compare](#) (const \_CharT \*\_\_lo1, const \_CharT \*\_\_hi1, const \_CharT \*\_\_lo2, const \_CharT \*\_\_hi2) const
- virtual long [do\\_hash](#) (const \_CharT \*\_\_lo, const \_CharT \*\_\_hi) const
- virtual [string\\_type do\\_transform](#) (const \_CharT \*\_\_lo, const \_CharT \*\_\_hi) const

### Static Protected Member Functions

- static \_\_c\_locale [\\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale [\\_S\\_get\\_c\\_locale](#) ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static \_\_c\_locale [\\_S\\_lc\\_ctype\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \*\_\_s)

### Protected Attributes

- \_\_c\_locale [\\_M\\_c\\_locale\\_collate](#)

### Friends

- class [locale::\\_Impl](#)

#### 5.424.1 Detailed Description

**template<typename \_CharT> class std::collate\_byname<\_CharT>**

class [collate\\_byname](#) [22.2.4.2].

Definition at line 794 of file locale\_classes.h.

### 5.424.2 Member Typedef Documentation

#### 5.424.2.1 `template<typename _CharT > typedef _CharT std::collate_byname< _CharT >::char_type`

Public typedefs.

Reimplemented from [std::collate< \\_CharT >](#).

Definition at line 799 of file locale\_classes.h.

#### 5.424.2.2 `template<typename _CharT > typedef basic_string<_CharT> std::collate_byname< _CharT >::string_type`

Public typedefs.

Reimplemented from [std::collate< \\_CharT >](#).

Definition at line 800 of file locale\_classes.h.

### 5.424.3 Member Function Documentation

#### 5.424.3.1 `template<typename _CharT> int std::collate< _CharT >::compare ( const _CharT * __lo1, const _CharT * __hi1, const _CharT * __lo2, const _CharT * __hi2 ) const [inline, inherited]`

Compare two strings.

This function compares two strings and returns the result by calling [collate::do\\_compare\(\)](#).

#### Parameters

*lo1* Start of string 1.

*hi1* End of string 1.

*lo2* Start of string 2.

*hi2* End of string 2.

#### Returns

1 if string1 > string2, -1 if string1 < string2, else 0.

Definition at line 676 of file locale\_classes.h.

**5.424.3.2** `template<typename _CharT> int std::collate<_CharT  
>::do_compare ( const _CharT * __lo1, const _CharT *  
__hi1, const _CharT * __lo2, const _CharT * __hi2 ) const  
[protected, virtual, inherited]`

Compare two strings.

This function is a hook for derived classes to change the value returned.

See also

[compare\(\)](#).

#### Parameters

*lo1* Start of string 1.

*hi1* End of string 1.

*lo2* Start of string 2.

*hi2* End of string 2.

#### Returns

1 if string1 > string2, -1 if string1 < string2, else 0.

Definition at line 136 of file locale\_classes.tcc.

References `std::basic_string<_CharT, _Traits, _Alloc>::c_str()`, `std::basic_string<_CharT, _Traits, _Alloc>::data()`, and `std::basic_string<_CharT, _Traits, _Alloc>::length()`.

**5.424.3.3** `template<typename _CharT> long std::collate<_CharT  
>::do_hash ( const _CharT * __lo, const _CharT * __hi ) const  
[protected, virtual, inherited]`

Return hash of a string.

This function computes and returns a hash on the input string. This function is a hook for derived classes to change the value returned.

#### Parameters

*lo* Start of string.

*hi* End of string.

**Returns**

Hash value.

Definition at line 231 of file locale\_classes.tcc.

**5.424.3.4** `template<typename _CharT> collate< _CharT >::string_type  
std::collate< _CharT >::do_transform( const _CharT * __lo, const  
_CharT * __hi ) const [protected, virtual, inherited]`

Transform string to comparable form.

This function is a hook for derived classes to change the value returned.

**Parameters**

*lo1* Start of string 1.

*hi1* End of string 1.

*lo2* Start of string 2.

*hi2* End of string 2.

**Returns**

1 if string1 > string2, -1 if string1 < string2, else 0.

Definition at line 175 of file locale\_classes.tcc.

References `std::basic_string< _CharT, _Traits, _Alloc >::append()`, `std::basic_string< _CharT, _Traits, _Alloc >::c_str()`, `std::basic_string< _CharT, _Traits, _Alloc >::data()`, `std::basic_string< _CharT, _Traits, _Alloc >::length()`, and `std::basic_string< _CharT, _Traits, _Alloc >::push_back()`.

**5.424.3.5** `template<typename _CharT> long std::collate< _CharT >::hash(  
const _CharT * __lo, const _CharT * __hi ) const [inline,  
inherited]`

Return hash of a string.

This function computes and returns a hash on the input string. It does so by returning [collate::do\\_hash\(\)](#).

**Parameters**

*lo* Start of string.



*hi* End of string.

### Returns

Hash value.

Definition at line 709 of file locale\_classes.h.

**5.424.3.6** `template<typename _CharT> string_type std::collate< _CharT  
>::transform ( const _CharT * __lo, const _CharT * __hi ) const  
[inline, inherited]`

Transform string to comparable form.

This function is a wrapper for strxfrm functionality. It takes the input string and returns a modified string that can be directly compared to other transformed strings. In the C locale, this function just returns a copy of the input string. In some other locales, it may replace two chars with one, change a char for another, etc. It does so by returning [collate::do\\_transform\(\)](#).

### Parameters

*lo* Start of string.

*hi* End of string.

### Returns

Transformed string\_type.

Definition at line 695 of file locale\_classes.h.

Referenced by `std::regex_traits< _Ch_type >::transform()`.

## 5.424.4 Member Data Documentation

**5.424.4.1** `template<typename _CharT> locale::id std::collate< _CharT >::id  
[static, inherited]`

Numpunct facet id.

Definition at line 635 of file locale\_classes.h.

The documentation for this class was generated from the following file:

- [locale\\_classes.h](#)

## 5.425 std::complex< \_Tp > Struct Template Reference

### Public Types

- typedef `_Tp` `value_type`

### Public Member Functions

- constexpr `complex` (const `_Tp` &\_\_r=\_Tp(), const `_Tp` &\_\_i=\_Tp())
- template<typename `_Up` >  
constexpr `complex` (const `complex`< `_Up` > &\_\_z)
- constexpr `complex` `__rep` () const
- constexpr `_Tp` `imag` () const
- void `imag` (`_Tp` \_\_val)
- `complex`< `_Tp` > & `operator*=` (const `_Tp` &)
- template<typename `_Up` >  
`complex`< `_Tp` > & `operator*=` (const `complex`< `_Up` > &)
- `complex`< `_Tp` > & `operator+=` (const `_Tp` &\_\_t)
- template<typename `_Up` >  
`complex`< `_Tp` > & `operator+=` (const `complex`< `_Up` > &)
- `complex`< `_Tp` > & `operator-=` (const `_Tp` &\_\_t)
- template<typename `_Up` >  
`complex`< `_Tp` > & `operator-=` (const `complex`< `_Up` > &)
- template<typename `_Up` >  
`complex`< `_Tp` > & `operator/=` (const `complex`< `_Up` > &)
- `complex`< `_Tp` > & `operator/=` (const `_Tp` &)
- template<typename `_Up` >  
`complex`< `_Tp` > & `operator=` (const `complex`< `_Up` > &)
- `complex`< `_Tp` > & `operator=` (const `_Tp` &)
- void `real` (`_Tp` \_\_val)
- constexpr `_Tp` `real` () const

### 5.425.1 Detailed Description

`template<typename _Tp> struct std::complex< _Tp >`

Template to represent complex numbers.

Specializations for float, double, and long double are part of the library. Results with any other type are not guaranteed.

### Parameters

***Tp*** Type of real and imaginary values.

Definition at line 124 of file `complex`.

### 5.425.2 Member Typedef Documentation

#### 5.425.2.1 `template<typename _Tp> typedef _Tp std::complex< _Tp >::value_type`

Value typedef.

Definition at line 127 of file complex.

### 5.425.3 Constructor & Destructor Documentation

#### 5.425.3.1 `template<typename _Tp> constexpr std::complex< _Tp >::complex ( const _Tp & __r = _Tp(), const _Tp & __i = _Tp() ) [inline]`

Default constructor. First parameter is x, second parameter is y. /// Unspecified parameters default to 0.

Definition at line 131 of file complex.

#### 5.425.3.2 `template<typename _Tp> template<typename _Up > constexpr std::complex< _Tp >::complex ( const complex< _Up > & __z ) [inline]`

Copy constructor.

Definition at line 138 of file complex.

### 5.425.4 Member Function Documentation

#### 5.425.4.1 `template<typename _Tp> complex<_Tp>& std::complex< _Tp >::operator+=( const _Tp & __t ) [inline]`

Add *t* to this complex number.

Definition at line 181 of file complex.

#### 5.425.4.2 `template<typename _Tp> complex<_Tp>& std::complex<_Tp>::operator-= ( const _Tp & __t ) [inline]`

Subtract  $t$  from this complex number.

Definition at line 190 of file `complex`.

The documentation for this struct was generated from the following file:

- [complex](#)

## 5.426 `std::condition_variable` Class Reference

[condition\\_variable](#)

### Public Types

- `typedef __native_type * native_handle_type`

### Public Member Functions

- `condition_variable` (const [condition\\_variable](#) &)
- `native_handle_type native_handle ()`
- `void notify_all ()`
- `void notify_one ()`
- `condition_variable & operator= (const condition_variable &)`
- `template<typename _Predicate > void wait (unique_lock< mutex > &__lock, _Predicate __p)`
- `void wait (unique_lock< mutex > &__lock)`
- `template<typename _Rep, typename _Period, typename _Predicate > bool wait_for (unique_lock< mutex > &__lock, const chrono::duration< _Rep, _Period > &__rtime, _Predicate __p)`
- `template<typename _Rep, typename _Period > cv_status wait_for (unique_lock< mutex > &__lock, const chrono::duration< _Rep, _Period > &__rtime)`
- `template<typename _Duration > cv_status wait_until (unique_lock< mutex > &__lock, const chrono::time_point< __clock_t, _Duration > &__atime)`
- `template<typename _Clock, typename _Duration > cv_status wait_until (unique_lock< mutex > &__lock, const chrono::time_point< _Clock, _Duration > &__atime)`

- `template<typename _Clock, typename _Duration, typename _Predicate >`  
`bool wait_until (unique_lock< mutex > &__lock, const chrono::time_point<`  
`_Clock, _Duration > &__atime, _Predicate __p)`

### 5.426.1 Detailed Description

#### `condition_variable`

Definition at line 59 of file `condition_variable`.

The documentation for this class was generated from the following file:

- `condition_variable`

## 5.427 `std::condition_variable_any` Class Reference

### `condition_variable_any`

#### Public Types

- `typedef condition_variable::native_handle_type native_handle_type`

#### Public Member Functions

- `condition_variable_any` (const `condition_variable_any` &)
- `native_handle_type native_handle` ()
- `void notify_all` ()
- `void notify_one` ()
- `condition_variable_any & operator=` (const `condition_variable_any` &)
- `template<typename _Lock, typename _Predicate >`  
`void wait` (\_Lock &\_\_lock, \_Predicate \_\_p)
- `template<typename _Lock >`  
`void wait` (\_Lock &\_\_lock)
- `template<typename _Lock, typename _Rep, typename _Period, typename _Predicate >`  
`bool wait_for` (\_Lock &\_\_lock, const `chrono::duration`< \_Rep, \_Period > &\_\_ -  
`_ptime, _Predicate __p)`
- `template<typename _Lock, typename _Rep, typename _Period >`  
`cv_status wait_for` (\_Lock &\_\_lock, const `chrono::duration`< \_Rep, \_Period >  
`&__ptime)`
- `template<typename _Lock, typename _Clock, typename _Duration >`  
`cv_status wait_until` (\_Lock &\_\_lock, const `chrono::time_point`< \_Clock, \_ -  
`Duration > &__atime)`

## 5.428 `std::conditional< _Cond, _Iftrue, _Iffalse >` Struct Template Reference 2459

- `template<typename _Lock , typename _Clock , typename _Duration , typename _Predicate > bool wait_until ( _Lock &__lock, const chrono::time\_point< _Clock, _Duration > &__atime, _Predicate __p)`

### 5.427.1 Detailed Description

[condition\\_variable\\_any](#)

Definition at line 168 of file `condition_variable`.

The documentation for this class was generated from the following file:

- [condition\\_variable](#)

## 5.428 `std::conditional< _Cond, _Iftrue, _Iffalse >` Struct Template Reference

`conditional`

### Public Types

- `typedef _Iftrue type`

### 5.428.1 Detailed Description

```
template<bool _Cond, typename _Iftrue, typename _Iffalse> struct
std::conditional< _Cond, _Iftrue, _Iffalse >
```

`conditional`

Definition at line 849 of file `type_traits`.

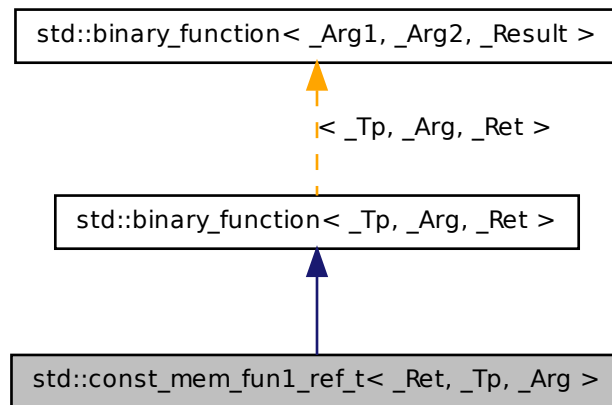
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.429 `std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >` Class Template Reference

One of the [adaptors for member /// pointers](#).

Inheritance diagram for `std::const_mem_fun1_ref_t<_Ret, _Tp, _Arg>`:



### Public Types

- `typedef _Tp first_argument_type`
- `typedef _Ret result_type`
- `typedef _Arg second_argument_type`

### Public Member Functions

- `const_mem_fun1_ref_t(_Ret(_Tp::*__pf)(_Arg) const)`
- `_Ret operator() (const _Tp &__r, _Arg __x) const`

#### 5.429.1 Detailed Description

`template<typename _Ret, typename _Tp, typename _Arg> class std::const_mem_fun1_ref_t<_Ret, _Tp, _Arg>`

One of the [adaptors for member /// pointers](#).

Definition at line 668 of file `stl_function.h`.

## 5.430 `std::const_mem_fun1_t<_Ret, _Tp, _Arg>` Class Template Reference 2461

### 5.429.2 Member Typedef Documentation

**5.429.2.1** `typedef _Tp std::binary_function< _Tp , _Arg , _Ret  
>::first_argument_type [inherited]`

`first_argument_type` is the type of the first argument

Definition at line 118 of file `stl_function.h`.

**5.429.2.2** `typedef _Ret std::binary_function< _Tp , _Arg , _Ret >::result_type  
[inherited]`

`result_type` is the return type

Definition at line 124 of file `stl_function.h`.

**5.429.2.3** `typedef _Arg std::binary_function< _Tp , _Arg , _Ret  
>::second_argument_type [inherited]`

`second_argument_type` is the type of the second argument

Definition at line 121 of file `stl_function.h`.

The documentation for this class was generated from the following file:

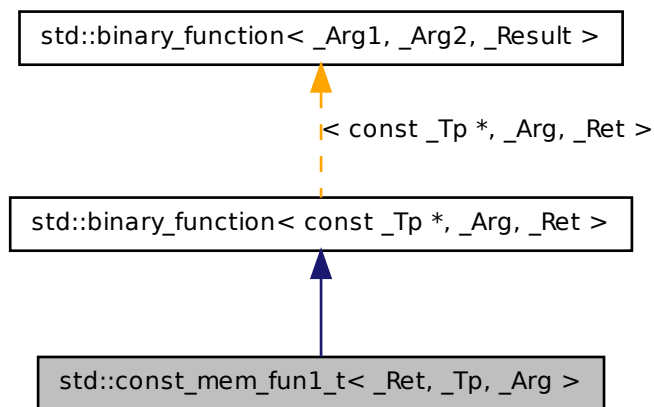
- [stl\\_function.h](#)

## 5.430 `std::const_mem_fun1_t<_Ret, _Tp, _Arg>` Class Template Reference

One of the [adaptors for member /// pointers](#).



Inheritance diagram for `std::const_mem_fun1_t< _Ret, _Tp, _Arg >`:



### Public Types

- `typedef const _Tp * first\_argument\_type`
- `typedef _Ret result\_type`
- `typedef _Arg second\_argument\_type`

### Public Member Functions

- `const_mem_fun1_t (_Ret(_Tp::*__pf)(_Arg) const)`
- `_Ret operator() (const _Tp *__p, _Arg __x) const`

#### 5.430.1 Detailed Description

`template<typename _Ret, typename _Tp, typename _Arg> class std::const_mem_fun1_t< _Ret, _Tp, _Arg >`

One of the [adaptors for member /// pointers](#).

Definition at line 632 of file `stl_function.h`.

### **5.430.2 Member Typedef Documentation**

**5.430.2.1** `typedef const _Tp * std::binary_function< const _Tp *, _Arg, _Ret >::first_argument_type [inherited]`

`first_argument_type` is the type of the first argument

Definition at line 118 of file `stl_function.h`.

**5.430.2.2** `typedef _Ret std::binary_function< const _Tp *, _Arg, _Ret >::result_type [inherited]`

`result_type` is the return type

Definition at line 124 of file `stl_function.h`.

**5.430.2.3** `typedef _Arg std::binary_function< const _Tp *, _Arg, _Ret >::second_argument_type [inherited]`

`second_argument_type` is the type of the second argument

Definition at line 121 of file `stl_function.h`.

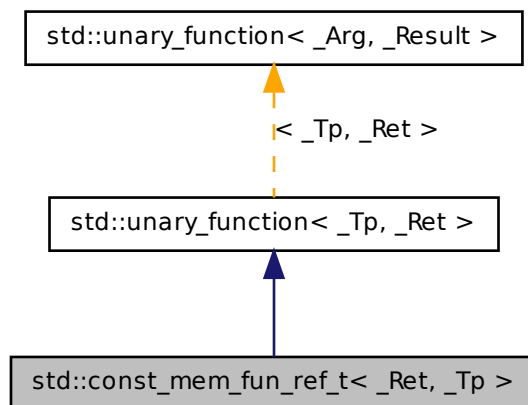
The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## **5.431 std::const\_mem\_fun\_ref\_t< \_Ret, \_Tp > Class Template Reference**

One of the [adaptors for member /// pointers](#).

Inheritance diagram for `std::const_mem_fun_ref_t<_Ret, _Tp>`:



### Public Types

- typedef `_Tp` [argument\\_type](#)
- typedef `_Ret` [result\\_type](#)

### Public Member Functions

- `const_mem_fun_ref_t` (`_Ret`(`_Tp`::\*\_\_pf)() const)
- `_Ret operator()` (const `_Tp` &\_\_r) const

#### 5.431.1 Detailed Description

**template<typename `_Ret`, typename `_Tp`> class `std::const_mem_fun_ref_t<_Ret, _Tp>`**

One of the [adaptors for member /// pointers](#).

Definition at line 596 of file `stl_function.h`.

### **5.431.2 Member Typedef Documentation**

#### **5.431.2.1** `typedef _Tp std::unary_function< _Tp , _Ret >::argument_type` `[inherited]`

`argument_type` is the type of the argument

Definition at line 105 of file `stl_function.h`.

#### **5.431.2.2** `typedef _Ret std::unary_function< _Tp , _Ret >::result_type` `[inherited]`

`result_type` is the return type

Definition at line 108 of file `stl_function.h`.

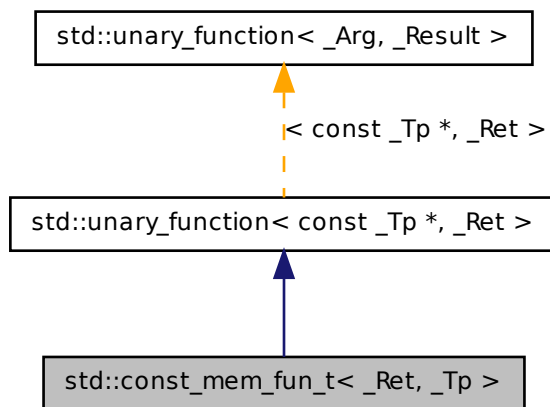
The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

### **5.432** `std::const_mem_fun_t<_Ret, _Tp>` Class Template Reference

One of the [adaptors for member /// pointers](#).

Inheritance diagram for `std::const_mem_fun_t<_Ret, _Tp>`:



### Public Types

- typedef `const _Tp *` [argument\\_type](#)
- typedef `_Ret` [result\\_type](#)

### Public Member Functions

- `const_mem_fun_t` (`_Ret(_Tp::*__pf)() const`)
- `_Ret operator()` (`const _Tp *__p`) `const`

#### 5.432.1 Detailed Description

`template<typename _Ret, typename _Tp> class std::const_mem_fun_t<_Ret, _Tp>`

One of the [adaptors for member /// pointers](#).

Definition at line 560 of file `stl_function.h`.

### 5.432.2 Member Typedef Documentation

#### 5.432.2.1 `typedef const _Tp * std::unary_function< const _Tp *, _Ret >::argument_type [inherited]`

`argument_type` is the type of the argument

Definition at line 105 of file `stl_function.h`.

#### 5.432.2.2 `typedef _Ret std::unary_function< const _Tp *, _Ret >::result_type [inherited]`

`result_type` is the return type

Definition at line 108 of file `stl_function.h`.

The documentation for this class was generated from the following file:

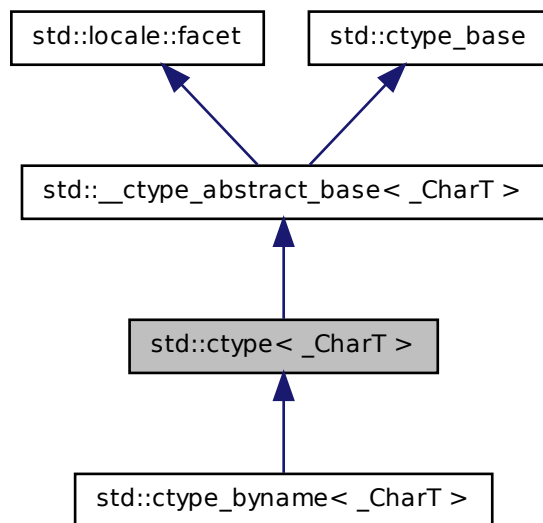
- [stl\\_function.h](#)

## 5.433 `std::ctype<_CharT>` Class Template Reference

Primary class template `ctype` facet.

This template class defines classification and conversion functions for character sets. It wraps `cctype` functionality. `Ctype` gets used by streams for many I/O operations.

Inheritance diagram for std::ctype< \_CharT >:



### Public Types

- typedef const int \* **\_\_to\_type**
- typedef \_CharT **char\_type**
- typedef \_\_ctype\_abstract\_base< \_CharT >::mask **mask**

### Public Member Functions

- **ctype** (size\_t \_\_refs=0)
- bool **is** (mask \_\_m, **char\_type** \_\_c) const
- const **char\_type** \* **is** (const **char\_type** \*\_\_lo, const **char\_type** \*\_\_hi, mask \*\_\_-vec) const
- const **char\_type** \* **narrow** (const **char\_type** \*\_\_lo, const **char\_type** \*\_\_hi, **char** \_\_default, **char** \*\_\_to) const
- **char** **narrow** (**char\_type** \_\_c, **char** \_\_default) const
- const **char\_type** \* **scan\_is** (mask \_\_m, const **char\_type** \*\_\_lo, const **char\_type** \*\_\_hi) const

- `const char_type * scan_not` (mask `__m`, `const char_type * __lo`, `const char_type * __hi`) `const`
- `char_type tolower` (`char_type __c`) `const`
- `const char_type * tolower` (`char_type * __lo`, `const char_type * __hi`) `const`
- `const char_type * toupper` (`char_type * __lo`, `const char_type * __hi`) `const`
- `char_type toupper` (`char_type __c`) `const`
- `const char * widen` (`const char * __lo`, `const char * __hi`, `char_type * __to`) `const`
- `char_type widen` (`char __c`) `const`

### Static Public Attributes

- static `const mask` **alnum**
- static `const mask` **alpha**
- static `const mask` **cntrl**
- static `const mask` **digit**
- static `const mask` **graph**
- static `locale::id` **id**
- static `const mask` **lower**
- static `const mask` **print**
- static `const mask` **punct**
- static `const mask` **space**
- static `const mask` **upper**
- static `const mask` **xdigit**

### Protected Member Functions

- virtual `bool do_is` (mask `__m`, `char_type __c`) `const`
- virtual `const char_type * do_is` (`const char_type * __lo`, `const char_type * __hi`, `mask * __vec`) `const`
- virtual `const char_type * do_narrow` (`const char_type * __lo`, `const char_type * __hi`, `char __dfault`, `char * __dest`) `const`
- virtual `char do_narrow` (`char_type`, `char __dfault`) `const`
- virtual `const char_type * do_scan_is` (mask `__m`, `const char_type * __lo`, `const char_type * __hi`) `const`
- virtual `const char_type * do_scan_not` (mask `__m`, `const char_type * __lo`, `const char_type * __hi`) `const`
- virtual `char_type do_tolower` (`char_type __c`) `const`
- virtual `const char_type * do_tolower` (`char_type * __lo`, `const char_type * __hi`) `const`
- virtual `const char_type * do_toupper` (`char_type * __lo`, `const char_type * __hi`) `const`
- virtual `char_type do_toupper` (`char_type __c`) `const`



- virtual const char \* [do\\_widen](#) (const char \*\_\_lo, const char \*\_\_hi, [char\\_type](#) \*\_\_dest) const
- virtual [char\\_type do\\_widen](#) (char \_\_c) const

#### Static Protected Member Functions

- static \_\_c\_locale [\\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale [\\_S\\_get\\_c\\_locale](#) ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static \_\_c\_locale [\\_S\\_lc\\_ctype\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \*\_\_s)

#### Friends

- class [locale::\\_Impl](#)

#### 5.433.1 Detailed Description

**template<typename \_CharT> class std::ctype<\_CharT>**

Primary class template ctype facet.

This template class defines classification and conversion functions for character sets. It wraps ctype functionality. Ctype gets used by streams for many I/O operations. This template provides the protected virtual functions the developer will have to replace in a derived class or specialization to make a working facet. The public functions that access them are defined in [\\_\\_ctype\\_abstract\\_base](#), to allow for implementation flexibility. See [ctype<wchar\\_t>](#) for an example. The functions are documented in [\\_\\_ctype\\_abstract\\_base](#).

Note: implementations are provided for all the protected virtual functions, but will likely not be useful.

Definition at line 606 of file locale\_facets.h.

#### 5.433.2 Member Typedef Documentation

**5.433.2.1 template<typename \_CharT> typedef \_CharT std::ctype<\_CharT>::char\_type**

Typedef for the template parameter.

Reimplemented from [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Definition at line 610 of file locale\_facets.h.

### 5.433.3 Member Function Documentation

**5.433.3.1** `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_is ( const char_type * __lo, const char_type * __hi, mask * __vec ) const [protected, virtual]`

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.

[do\\_is\(\)](#) is a hook for a derived facet to change the behavior of classifying. [do\\_is\(\)](#) must always return the same result for the same input.

#### Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- vec* Pointer to an array of mask storage.

#### Returns

*hi*.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**5.433.3.2** `template<typename _CharT> virtual bool std::ctype<_CharT>::do_is ( mask __m, char_type __c ) const [protected, virtual]`

Test char\_type classification.

This function finds a mask M for *c* and compares it to mask *m*.

[do\\_is\(\)](#) is a hook for a derived facet to change the behavior of classifying. [do\\_is\(\)](#) must always return the same result for the same input.

#### Parameters

- c* The char\_type to find the mask of.

*m* The mask to compare against.

### Returns

$(M \& m) \neq 0$ .

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**5.433.3.3** `template<typename _CharT> virtual char std::ctype<_CharT>::do_narrow ( char_type, char __dfault ) const [protected, virtual]`

Narrow `char_type` to `char`.

This virtual function converts the argument to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead.

[do\\_narrow\(\)](#) is a hook for a derived facet to change the behavior of narrowing. [do\\_narrow\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

### Parameters

*c* The `char_type` to convert.

*dfault* `Char` to return if conversion fails.

### Returns

The converted `char`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Referenced by `std::ctype<char>::narrow()`.

**5.433.3.4** `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_narrow ( const char_type * __lo, const char_type * __hi, char __dfault, char * __dest ) const [protected, virtual]`

Narrow `char_type` array to `char`.

This virtual function converts each `char_type` in the range `[lo,hi)` to `char` using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, *dfault* is used instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- dfault* Char to use if conversion fails.
- to* Pointer to the destination array.

#### Returns

*hi*.

Implements `std::__ctype_abstract_base<_CharT>`.

**5.433.3.5** `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_scan_is( mask __m, const char_type* __lo, const char_type* __hi ) const` [`protected`, `virtual`]

Find `char_type` matching mask.

This function searches for and returns the first `char_type` `c` in `[lo,hi)` for which `is(m,c)` is true.

`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.

#### Parameters

- m* The mask to compare against.
- lo* Pointer to start of range.
- hi* Pointer to end of range.

#### Returns

Pointer to a matching `char_type` if found, else *hi*.

Implements `std::__ctype_abstract_base<_CharT>`.

**5.433.3.6** `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_scan_not ( mask __m, const char_type * __lo, const char_type * __hi ) const [protected, virtual]`

Find `char_type` not matching mask.

This function searches for and returns a pointer to the first `char_type` `c` of `[lo,hi)` for which `is(m,c)` is false.

`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.

#### Parameters

*m* The mask to compare against.

*lo* Pointer to start of range.

*hi* Pointer to end of range.

#### Returns

Pointer to a non-matching `char_type` if found, else *hi*.

Implements `std::__ctype_abstract_base<_CharT>`.

**5.433.3.7** `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_tolower ( char_type * __lo, const char_type * __hi ) const [protected, virtual]`

Convert array to lowercase.

This virtual function converts each `char_type` in the range `[lo,hi)` to lowercase if possible. Other elements remain untouched.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

#### Parameters

*lo* Pointer to start of range.

*hi* Pointer to end of range.

#### Returns

*hi*.

Implements `std::__ctype_abstract_base<_CharT>`.

### 5.433.3.8 `template<typename _CharT> virtual char_type std::ctype< _CharT >::do_tolower ( char_type ) const [protected, virtual]`

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

#### Parameters

*c* The `char_type` to convert.

#### Returns

The lowercase `char_type` if convertible, else *c*.

Implements `std::__ctype_abstract_base< _CharT >`.

Referenced by `std::ctype< char >::tolower()`.

### 5.433.3.9 `template<typename _CharT> virtual const char_type* std::ctype< _CharT >::do_toupper ( char_type * __lo, const char_type * __hi ) const [protected, virtual]`

Convert array to uppercase.

This virtual function converts each `char_type` in the range [lo,hi) to uppercase if possible. Other elements remain untouched.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

#### Parameters

*lo* Pointer to start of range.

*hi* Pointer to end of range.

#### Returns

*hi*.

Implements `std::__ctype_abstract_base< _CharT >`.

**5.433.3.10** `template<typename _CharT> virtual char_type std::ctype<_CharT>::do_toupper ( char_type ) const [protected, virtual]`

Convert to uppercase.

This virtual function converts the `char_type` argument to uppercase if possible. If not possible (for example, `'2'`), returns the argument.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

#### Parameters

*c* The `char_type` to convert.

#### Returns

The uppercase `char_type` if convertible, else *c*.

Implements `std::__ctype_abstract_base<_CharT>`.

Referenced by `std::ctype<char>::toupper()`.

**5.433.3.11** `template<typename _CharT> virtual char_type std::ctype<_CharT>::do_widen ( char ) const [protected, virtual]`

Widen char.

This virtual function converts the `char` to `char_type` using the simplest reasonable transformation.

`do_widen()` is a hook for a derived facet to change the behavior of widening. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

*c* The `char` to convert.

#### Returns

The converted `char_type`

Implements `std::__ctype_abstract_base<_CharT>`.

Referenced by `std::ctype<char>::widen()`.

**5.433.3.12** `template<typename _CharT> virtual const char* std::ctype<_CharT>::do_widen ( const char * __lo, const char * __hi, char_type * __dest ) const [protected, virtual]`

Widen char array.

This function converts each char in the input to char\_type using the simplest reasonable transformation.

[do\\_widen\(\)](#) is a hook for a derived facet to change the behavior of widening. [do\\_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See [codecvt](#) for that.

#### Parameters

- lo* Pointer to start range.
- hi* Pointer to end of range.
- to* Pointer to the destination array.

#### Returns

*hi*.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**5.433.3.13** `template<typename _CharT> bool std::__ctype_abstract_base<_CharT>::is ( mask __m, char_type __c ) const [inline, inherited]`

Test char\_type classification.

This function finds a mask *M* for *c* and compares it to mask *m*. It does so by returning the value of [ctype<char\\_type>::do\\_is\(\)](#).

#### Parameters

- c* The char\_type to compare the mask of.
- m* The mask to compare against.

#### Returns

(*M* & *m*) != 0.



Definition at line 163 of file locale\_facets.h.

Referenced by std::regex\_traits<\_Ch\_type>::isctype(), and std::basic\_istream<\_CharT, \_Traits>::sentry::sentry().

**5.433.3.14** `template<typename _CharT> const char_type*  
std::__ctype_abstract_base<_CharT>::is ( const char_type *  
__lo, const char_type * __hi, mask * __vec ) const [inline,  
inherited]`

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of `ctype<char_type>::do_is()`.

#### Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- vec* Pointer to an array of mask storage.

#### Returns

*hi*.

Definition at line 180 of file locale\_facets.h.

**5.433.3.15** `template<typename _CharT> char std::__ctype_abstract_base<  
_CharT>::narrow ( char_type __c, char __dfault ) const  
[inline, inherited]`

Narrow char\_type to char.

This function converts the char\_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning `ctype<char_type>::do_narrow(c)`.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

- c* The char\_type to convert.
- dfault* Char to return if conversion fails.

**Returns**

The converted char.

Definition at line 325 of file locale\_facets.h.

Referenced by std::time\_put<\_CharT, \_OutIter>::put().

**5.433.3.16** `template<typename _CharT> const char_type*  
std::__ctype_abstract_base<_CharT>::narrow ( const char_type  
* __lo, const char_type * __hi, char __dfault, char * __to ) const  
[inline, inherited]`

Narrow array to char array.

This function converts each char\_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char\_type in the input that cannot be converted, *dfault* is used instead. It does so by returning ctype<char\_type>::do\_narrow(lo, hi, dfault, to).

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

*lo* Pointer to start of range.

*hi* Pointer to end of range.

*dfault* Char to use if conversion fails.

*to* Pointer to the destination array.

**Returns**

*hi*.

Definition at line 347 of file locale\_facets.h.

**5.433.3.17** `template<typename _CharT> const char_type*  
std::__ctype_abstract_base<_CharT>::scan_is ( mask __m,  
const char_type * __lo, const char_type * __hi ) const [inline,  
inherited]`

Find char\_type matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char\_type>::do\_scan\_is().

**Parameters**

*m* The mask to compare against.

*lo* Pointer to start of range.

*hi* Pointer to end of range.

**Returns**

Pointer to matching char\_type if found, else *hi*.

Definition at line 196 of file locale\_facets.h.

```
5.433.3.18 template<typename _CharT> const char_type*
std::__ctype_abstract_base<_CharT>::scan_not (mask __m,
const char_type * __lo, const char_type * __hi) const [inline,
inherited]
```

Find char\_type not matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is false. It does so by returning [ctype<char\\_type>::do\\_scan\\_not\(\)](#).

**Parameters**

*m* The mask to compare against.

*lo* Pointer to first char in range.

*hi* Pointer to end of range.

**Returns**

Pointer to non-matching char if found, else *hi*.

Definition at line 212 of file locale\_facets.h.

```
5.433.3.19 template<typename _CharT> const char_type*
std::__ctype_abstract_base<_CharT>::tolower (char_type *
__lo, const char_type * __hi) const [inline, inherited]
```

Convert array to lowercase.

This function converts each char\_type in the range [lo,hi) to lowercase if possible. Other elements remain untouched. It does so by returning [ctype<char\\_type>::do\\_toupper\(lo, hi\)](#).

**Parameters**

*lo* Pointer to start of range.

*hi* Pointer to end of range.

**Returns**

*hi*.

Definition at line 270 of file locale\_facets.h.

**5.433.3.20** `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::tolower ( char_type __c ) const [inline, inherited]`

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char\_type>::do\_toupper(c).

**Parameters**

*c* The char\_type to convert.

**Returns**

The lowercase char\_type if convertible, else *c*.

Definition at line 255 of file locale\_facets.h.

**5.433.3.21** `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::toupper ( char_type * __lo, const char_type * __hi ) const [inline, inherited]`

Convert array to uppercase.

This function converts each char\_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning ctype<char\_type>::do\_toupper(lo, hi).

**Parameters**

*lo* Pointer to start of range.

*hi* Pointer to end of range.

#### Returns

*hi*.

Definition at line 241 of file `locale_facets.h`.

**5.433.3.22** `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::toupper ( char_type __c ) const [inline, inherited]`

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_toupper()`.

#### Parameters

*c* The `char_type` to convert.

#### Returns

The uppercase `char_type` if convertible, else *c*.

Definition at line 226 of file `locale_facets.h`.

**5.433.3.23** `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::widen ( char __c ) const [inline, inherited]`

Widen `char` to `char_type`.

This function converts the `char` argument to `char_type` using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

*c* The `char` to convert.

#### Returns

The converted `char_type`.

Definition at line 287 of file locale\_facets.h.

Referenced by std::money\_get<\_CharT, \_InIter>::do\_get(), std::time\_put<\_CharT, \_OutIter>::do\_put(), std::money\_put<\_CharT, \_OutIter>::do\_put(), std::regex\_traits<\_Ch\_type>::isctype(), and std::operator<<().

**5.433.3.24** `template<typename _CharT> const char* std::__ctype_abstract_base<_CharT>::widen ( const char * __lo, const char * __hi, char_type * __to ) const [inline, inherited]`

Widen array to char\_type.

This function converts each char in the input to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- to* Pointer to the destination array.

#### Returns

*hi*.

Definition at line 306 of file locale\_facets.h.

### 5.433.4 Member Data Documentation

**5.433.4.1** `template<typename _CharT> locale::id std::ctype<_CharT>::id [static]`

The facet id for ctype<char\_type>

Definition at line 614 of file locale\_facets.h.

The documentation for this class was generated from the following file:

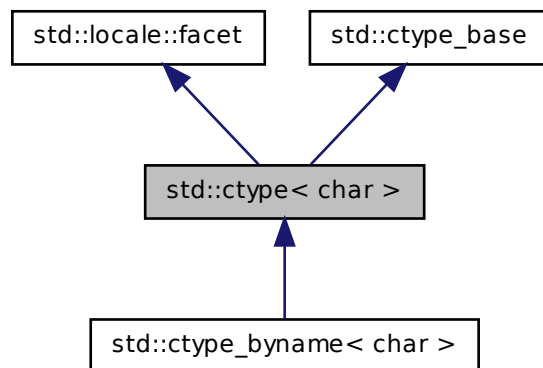
- [locale\\_facets.h](#)

### 5.434 `std::ctype< char >` Class Template Reference

The `ctype<char>` specialization.

This class defines classification and conversion functions for the `char` type. It gets used by `char` streams for many I/O operations. The `char` specialization provides a number of optimizations as well.

Inheritance diagram for `std::ctype< char >`:



#### Public Types

- typedef const int \* **\_\_to\_type**
- typedef char `char_type`
- typedef unsigned short **mask**

#### Public Member Functions

- `ctype` (const mask \* \_\_table=0, bool \_\_del=false, size\_t \_\_refs=0)
- `ctype` (\_\_c\_locale \_\_cloc, const mask \* \_\_table=0, bool \_\_del=false, size\_t \_\_refs=0)
- const char \* `is` (const char \* \_\_lo, const char \* \_\_hi, mask \* \_\_vec) const
- bool `is` (mask \_\_m, char \_\_c) const
- char `narrow` (`char_type` \_\_c, char \_\_dfault) const

- `const char_type * narrow` (`const char_type *__lo`, `const char_type *__hi`, `char __dfault`, `char *__to`) `const`
- `const char * scan_is` (`mask __m`, `const char *__lo`, `const char *__hi`) `const`
- `const char * scan_not` (`mask __m`, `const char *__lo`, `const char *__hi`) `const`
- `const mask * table` () `const throw ()`
- `char_type tolower` (`char_type __c`) `const`
- `const char_type * tolower` (`char_type *__lo`, `const char_type *__hi`) `const`
- `const char_type * toupper` (`char_type *__lo`, `const char_type *__hi`) `const`
- `char_type toupper` (`char_type __c`) `const`
- `char_type widen` (`char __c`) `const`
- `const char * widen` (`const char *__lo`, `const char *__hi`, `char_type *__to`) `const`

### Static Public Member Functions

- `static const mask * classic_table` () `throw ()`

### Static Public Attributes

- `static const mask alnum`
- `static const mask alpha`
- `static const mask cntrl`
- `static const mask digit`
- `static const mask graph`
- `static locale::id id`
- `static const mask lower`
- `static const mask print`
- `static const mask punct`
- `static const mask space`
- `static const size_t table_size`
- `static const mask upper`
- `static const mask xdigit`

### Protected Member Functions

- `virtual ~ctype` ()
- `virtual char do_narrow` (`char_type __c`, `char`) `const`
- `virtual const char_type * do_narrow` (`const char_type *__lo`, `const char_type *__hi`, `char`, `char *__dest`) `const`
- `virtual char_type do_tolower` (`char_type`) `const`
- `virtual const char_type * do_tolower` (`char_type *__lo`, `const char_type *__hi`) `const`
- `virtual char_type do_toupper` (`char_type`) `const`



- virtual const `char_type * do_toupper (char_type *__lo, const char_type *__hi)` const
- virtual `char_type do_widen (char __c)` const
- virtual const char \* `do_widen (const char *__lo, const char *__hi, char_type *__dest)` const

#### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static const char \* `_S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

#### Protected Attributes

- `__c_locale _M_c_locale_ctype`
- bool `_M_del`
- char `_M_narrow [1+static_cast< unsigned char >(-1)]`
- char `_M_narrow_ok`
- const mask \* `_M_table`
- `__to_type _M_tolower`
- `__to_type _M_toupper`
- char `_M_widen [1+static_cast< unsigned char >(-1)]`
- char `_M_widen_ok`

#### Friends

- class `locale::_Impl`

#### 5.434.1 Detailed Description

`template<> class std::ctype< char >`

The `ctype<char>` specialization.

This class defines classification and conversion functions for the char type. It gets used by char streams for many I/O operations. The char specialization provides a number of optimizations as well.

Definition at line 675 of file `locale_facets.h`.

### 5.434.2 Member Typedef Documentation

#### 5.434.2.1 `typedef char std::ctype< char >::char_type`

Typedef for the template parameter `char`.

Definition at line 680 of file `locale_facets.h`.

### 5.434.3 Constructor & Destructor Documentation

#### 5.434.3.1 `std::ctype< char >::ctype ( const mask * __table = 0, bool __del = false, size_t __refs = 0 ) [explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

##### Parameters

*table* If non-zero, table is used as the per-char mask. Else [classic\\_table\(\)](#) is used.

*del* If true, passes ownership of table to this facet.

*refs* Passed to the base facet class.

#### 5.434.3.2 `std::ctype< char >::ctype ( __c_locale __cloc, const mask * __table = 0, bool __del = false, size_t __refs = 0 ) [explicit]`

Constructor performs static initialization.

This constructor is used to construct the initial C locale facet.

##### Parameters

*cloc* Handle to C locale data.

*table* If non-zero, table is used as the per-char mask.

*del* If true, passes ownership of table to this facet.

*refs* Passed to the base facet class.

### 5.434.3.3 virtual std::ctype< char >::~~ctype ( ) [protected, virtual]

Destructor.

This function deletes [table\(\)](#) if *del* was true in the constructor.

### 5.434.4 Member Function Documentation

#### 5.434.4.1 static const mask\* std::ctype< char >::classic\_table ( ) throw () [static]

Returns a pointer to the C locale mask table.

#### 5.434.4.2 virtual char std::ctype< char >::do\_narrow ( char\_type \_\_c, char ) const [inline, protected, virtual]

Narrow char.

This virtual function converts the char to char using the simplest reasonable transformation. If the conversion fails, *dfault* is returned instead. For an underived [ctype<char>](#) facet, *c* will be returned unchanged.

[do\\_narrow\(\)](#) is a hook for a derived facet to change the behavior of narrowing. [do\\_narrow\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See [codecvt](#) for that.

#### Parameters

*c* The char to convert.

*dfault* Char to return if conversion fails.

#### Returns

The converted char.

Definition at line 1125 of file [locale\\_facets.h](#).

#### 5.434.4.3 virtual const char\_type\* std::ctype< char >::do\_narrow ( const char\_type \* \_\_lo, const char\_type \* \_\_hi, char, char \* \_\_dest ) const [inline, protected, virtual]

Narrow char array to char array.

This virtual function converts each char in the range [lo,hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *dfault* is used instead. For an underived `ctype<char>` facet, the argument will be copied unchanged.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- dfault* Char to use if conversion fails.
- to* Pointer to the destination array.

#### Returns

*hi*.

Definition at line 1151 of file `locale_facets.h`.

**5.434.4.4** `virtual const char_type* std::ctype< char >::do_tolower ( char_type * __lo, const char_type * __hi ) const [protected, virtual]`

Convert array to lowercase.

This virtual function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

#### Parameters

- lo* Pointer to first char in range.
- hi* Pointer to end of range.

#### Returns

*hi*.

**5.434.4.5 virtual char\_type std::ctype< char >::do\_tolower ( char\_type )  
const [protected, virtual]**

Convert to lowercase.

This virtual function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

[do\\_tolower\(\)](#) is a hook for a derived facet to change the behavior of lowercasing. [do\\_tolower\(\)](#) must always return the same result for the same input.

**Parameters**

*c* The char to convert.

**Returns**

The lowercase char if convertible, else *c*.

**5.434.4.6 virtual const char\_type\* std::ctype< char >::do\_toupper ( char\_type  
\* \_\_lo, const char\_type \* \_\_hi ) const [protected, virtual]**

Convert array to uppercase.

This virtual function converts each char in the range [lo,hi) to uppercase if possible. Other chars remain untouched.

[do\\_toupper\(\)](#) is a hook for a derived facet to change the behavior of uppercasing. [do\\_toupper\(\)](#) must always return the same result for the same input.

**Parameters**

*lo* Pointer to start of range.

*hi* Pointer to end of range.

**Returns**

*hi*.

**5.434.4.7 virtual char\_type std::ctype< char >::do\_toupper ( char\_type )  
const [protected, virtual]**

Convert to uppercase.

This virtual function converts the `char` argument to uppercase if possible. If not possible (for example, `'2'`), returns the argument.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

#### Parameters

`c` The `char` to convert.

#### Returns

The uppercase `char` if convertible, else `c`.

#### 5.434.4.8 `virtual char_type std::ctype< char >::do_widen ( char __c ) const [inline, protected, virtual]`

Widen `char`.

This virtual function converts the `char` to `char_type` using the simplest reasonable transformation. For an undervived `ctype<char>` facet, the argument will be returned unchanged.

`do_widen()` is a hook for a derived facet to change the behavior of widening. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

`c` The `char` to convert.

#### Returns

The converted character.

Definition at line 1076 of file `locale_facets.h`.

#### 5.434.4.9 `virtual const char* std::ctype< char >::do_widen ( const char * __lo, const char * __hi, char_type * __dest ) const [inline, protected, virtual]`

Widen `char` array.

This function converts each char in the range [lo,hi) to char using the simplest reasonable transformation. For an underived `ctype<char>` facet, the argument will be copied unchanged.

`do_widen()` is a hook for a derived facet to change the behavior of widening. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- to* Pointer to the destination array.

#### Returns

*hi*.

Definition at line 1099 of file `locale_facets.h`.

**5.434.4.10** `const char * std::ctype< char >::is ( const char * __lo, const char * __hi, mask * __vec ) const [inline]`

Return a mask array.

This function finds the mask for each char in the range [lo, hi) and successively writes it to vec. vec must have as many elements as the char array.

#### Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- vec* Pointer to an array of mask storage.

#### Returns

*hi*.

Definition at line 48 of file `ctype_inline.h`.

**5.434.4.11** `bool std::ctype< char >::is ( mask __m, char __c ) const [inline]`

Test char classification.

This function compares the mask table[c] to *m*.

#### Parameters

*c* The char to compare the mask of.

*m* The mask to compare against.

#### Returns

True if m & table[c] is true, false otherwise.

Definition at line 43 of file ctype\_inline.h.

**5.434.4.12** `char std::ctype< char >::narrow ( char_type __c, char __dfault )  
const [inline]`

Narrow char.

This function converts the char to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. For an undervied `ctype<char>` facet, *c* will be returned unchanged.

This function works as if it returns `ctype<char>::do_narrow(c)`. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

*c* The char to convert.

*dfault* Char to return if conversion fails.

#### Returns

The converted character.

Definition at line 924 of file locale\_facets.h.

References `std::ctype< _CharT >::do_narrow()`.

**5.434.4.13** `const char_type* std::ctype< char >::narrow ( const char_type *  
__lo, const char_type * __hi, char __dfault, char * __to ) const  
[inline]`



Narrow char array.

This function converts each char in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *dfault* is used instead. For an underived `ctype<char>` facet, the argument will be copied unchanged.

This function works as if it returns `ctype<char>::do_narrow(lo, hi, dfault, to)`. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- dfault* Char to use if conversion fails.
- to* Pointer to the destination array.

#### Returns

*hi*.

Definition at line 957 of file `locale_facets.h`.

References `std::ctype<_CharT>::do_narrow()`.

**5.434.4.14** `const char * std::ctype< char >::scan_is ( mask __m, const char *  
__lo, const char * __hi ) const [inline]`

Find char matching a mask.

This function searches for and returns the first char in [lo,hi) for which `is(m,char)` is true.

#### Parameters

- m* The mask to compare against.
- lo* Pointer to start of range.
- hi* Pointer to end of range.

#### Returns

Pointer to a matching char if found, else *hi*.

Definition at line 57 of file `ctype_inline.h`.

**5.434.4.15** `const char * std::ctype< char >::scan_not ( mask __m, const char * __lo, const char * __hi ) const [inline]`

Find char not matching a mask.

This function searches for and returns a pointer to the first char in [lo,hi) for which is(m,char) is false.

#### Parameters

*m* The mask to compare against.

*lo* Pointer to start of range.

*hi* Pointer to end of range.

#### Returns

Pointer to a non-matching char if found, else *hi*.

Definition at line 67 of file ctype\_inline.h.

**5.434.4.16** `const mask* std::ctype< char >::table ( ) const throw () [inline]`

Returns a pointer to the mask table provided to the constructor, or /// the default from [classic\\_table\(\)](#) if none was provided.

Definition at line 975 of file locale\_facets.h.

**5.434.4.17** `char_type std::ctype< char >::tolower ( char_type __c ) const [inline]`

Convert to lowercase.

This function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

[tolower\(\)](#) acts as if it returns `ctype<char>::do_tolower(c)`. [do\\_tolower\(\)](#) must always return the same result for the same input.

#### Parameters

*c* The char to convert.

**Returns**

The lowercase char if convertible, else *c*.

Definition at line 829 of file locale\_facets.h.

References std::ctype< \_CharT >::do\_tolower().

**5.434.4.18** `const char_type* std::ctype< char >::tolower ( char_type * __lo,  
const char_type * __hi ) const [inline]`

Convert array to lowercase.

This function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

`tolower()` acts as if it returns `ctype<char>:: do_tolower(lo, hi)`. `do_tolower()` must always return the same result for the same input.

**Parameters**

*lo* Pointer to first char in range.

*hi* Pointer to end of range.

**Returns**

*hi*.

Definition at line 846 of file locale\_facets.h.

References std::ctype< \_CharT >::do\_tolower().

**5.434.4.19** `const char_type* std::ctype< char >::toupper ( char_type * __lo,  
const char_type * __hi ) const [inline]`

Convert array to uppercase.

This function converts each char in the range [lo,hi) to uppercase if possible. Other chars remain untouched.

`toupper()` acts as if it returns `ctype<char>:: do_toupper(lo, hi)`. `do_toupper()` must always return the same result for the same input.

**Parameters**

*lo* Pointer to first char in range.

*hi* Pointer to end of range.

### Returns

*hi*.

Definition at line 813 of file locale\_facets.h.

References std::ctype< \_CharT >::do\_toupper().

#### 5.434.4.20 char\_type std::ctype< char >::toupper ( char\_type \_\_c ) const [inline]

Convert to uppercase.

This function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

[toupper\(\)](#) acts as if it returns ctype<char>::do\_toupper(c). [do\\_toupper\(\)](#) must always return the same result for the same input.

### Parameters

*c* The char to convert.

### Returns

The uppercase char if convertible, else *c*.

Definition at line 796 of file locale\_facets.h.

References std::ctype< \_CharT >::do\_toupper().

#### 5.434.4.21 char\_type std::ctype< char >::widen ( char \_\_c ) const [inline]

Widen char.

This function converts the char to char\_type using the simplest reasonable transformation. For an undervived [ctype<char>](#) facet, the argument will be returned unchanged.

This function works as if it returns ctype<char>::do\_widen(c). [do\\_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

*c* The char to convert.

**Returns**

The converted character.

Definition at line 866 of file locale\_facets.h.

References std::ctype< \_CharT >::do\_widen().

**5.434.4.22** `const char* std::ctype< char >::widen ( const char * __lo, const char * __hi, char_type * __to ) const [inline]`

Widen char array.

This function converts each char in the input to char using the simplest reasonable transformation. For an underived `ctype<char>` facet, the argument will be copied unchanged.

This function works as if it returns `ctype<char>::do_widen(c)`. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

*lo* Pointer to first char in range.

*hi* Pointer to end of range.

*to* Pointer to the destination array.

**Returns**

*hi*.

Definition at line 893 of file locale\_facets.h.

References std::ctype< \_CharT >::do\_widen().

**5.434.5 Member Data Documentation**

**5.434.5.1** `locale::id std::ctype< char >::id [static]`

The facet id for `ctype<char>`

Definition at line 697 of file locale\_facets.h.

**5.434.5.2 `const size_t std::ctype< char >::table_size` [static]**

The size of the mask table. It is `SCHAR_MAX + 1`.

Definition at line 699 of file `locale_facets.h`.

The documentation for this class was generated from the following files:

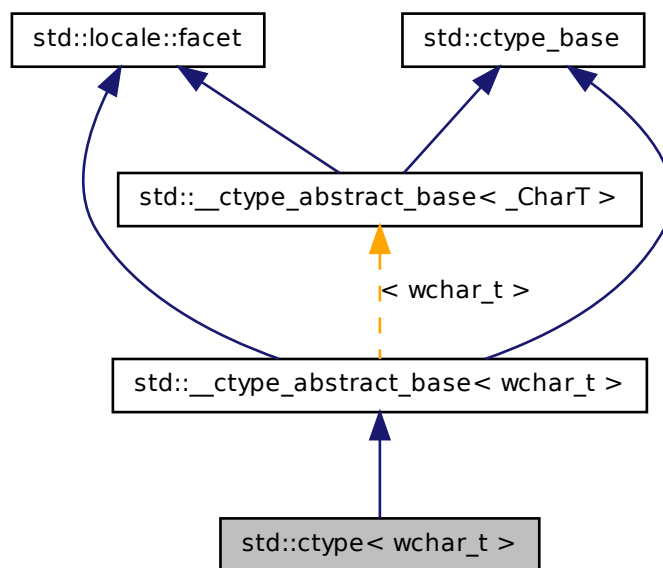
- [locale\\_facets.h](#)
- [ctype\\_inline.h](#)

**5.435 `std::ctype< wchar_t >` Class Template Reference**

The `ctype<wchar_t>` specialization.

This class defines classification and conversion functions for the `wchar_t` type. It gets used by `wchar_t` streams for many I/O operations. The `wchar_t` specialization provides a number of optimizations as well.

Inheritance diagram for `std::ctype< wchar_t >`:



### Public Types

- `typedef const int * __to_type`
- `typedef wctype_t __wmask_type`
- `typedef wchar_t char_type`
- `typedef unsigned short mask`

### Public Member Functions

- `ctype` (`size_t __refs=0`)
- `ctype` (`__c_locale __cloc, size_t __refs=0`)
- `const char_type * is` (`const char_type * __lo, const char_type * __hi, mask * __vec`) `const`
- `bool is` (`mask __m, char_type __c`) `const`
- `char narrow` (`char_type __c, char __dfault`) `const`

- `const char_type * narrow` (`const char_type *__lo`, `const char_type *__hi`, `char __default`, `char *__to`) `const`
- `const char_type * scan_is` (`mask __m`, `const char_type *__lo`, `const char_type *__hi`) `const`
- `const char_type * scan_not` (`mask __m`, `const char_type *__lo`, `const char_type *__hi`) `const`
- `char_type tolower` (`char_type __c`) `const`
- `const char_type * tolower` (`char_type *__lo`, `const char_type *__hi`) `const`
- `char_type toupper` (`char_type __c`) `const`
- `const char_type * toupper` (`char_type *__lo`, `const char_type *__hi`) `const`
- `const char * widen` (`const char *__lo`, `const char *__hi`, `char_type *__to`) `const`
- `char_type widen` (`char __c`) `const`

### Static Public Attributes

- static `const mask` **alnum**
- static `const mask` **alpha**
- static `const mask` **cntrl**
- static `const mask` **digit**
- static `const mask` **graph**
- static `locale::id` **id**
- static `const mask` **lower**
- static `const mask` **print**
- static `const mask` **punct**
- static `const mask` **space**
- static `const mask` **upper**
- static `const mask` **xdigit**

### Protected Member Functions

- virtual `~ctype` ()
- `__wmask_type _M_convert_to_wmask` (`const mask __m`) `const throw ()`
- `void _M_initialize_ctype` () `throw ()`
- virtual `const char_type * do_is` (`const char_type *__lo`, `const char_type *__hi`, `mask *__vec`) `const`
- virtual `bool do_is` (`mask __m`, `char_type __c`) `const`
- virtual `const char_type * do_narrow` (`const char_type *__lo`, `const char_type *__hi`, `char __default`, `char *__dest`) `const`
- virtual `char do_narrow` (`char_type`, `char __default`) `const`
- virtual `const char_type * do_scan_is` (`mask __m`, `const char_type *__lo`, `const char_type *__hi`) `const`
- virtual `const char_type * do_scan_not` (`mask __m`, `const char_type *__lo`, `const char_type *__hi`) `const`



- virtual `char_type do_tolower (char_type) const`
- virtual const `char_type * do_tolower (char_type *__lo, const char_type *__hi) const`
- virtual const `char_type * do_toupper (char_type *__lo, const char_type *__hi) const`
- virtual `char_type do_toupper (char_type) const`
- virtual const char \* `do_widen (const char *__lo, const char *__hi, char_type *__dest) const`
- virtual `char_type do_widen (char) const`

### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static const char \* `_S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

### Protected Attributes

- mask `_M_bit [16]`
- `__c_locale _M_c_locale_ctype`
- char `_M_narrow [128]`
- bool `_M_narrow_ok`
- `wint_t _M_widen [1+static_cast< unsigned char >(-1)]`
- `__wmask_type _M_wmask [16]`

### Friends

- class `locale::_Impl`

### 5.435.1 Detailed Description

`template<> class std::ctype< wchar_t >`

The `ctype<wchar_t>` specialization.

This class defines classification and conversion functions for the `wchar_t` type. It gets used by `wchar_t` streams for many I/O operations. The `wchar_t` specialization provides a number of optimizations as well. `ctype<wchar_t>` inherits its public methods from `__ctype_abstract_base<wchar_t>`.

Definition at line 1176 of file `locale_facets.h`.

## 5.435.2 Member Typedef Documentation

### 5.435.2.1 `typedef wchar_t std::ctype< wchar_t >::char_type`

Typedef for the template parameter `wchar_t`.

Reimplemented from `std::__ctype_abstract_base< wchar_t >`.

Definition at line 1181 of file `locale_facets.h`.

## 5.435.3 Constructor & Destructor Documentation

### 5.435.3.1 `std::ctype< wchar_t >::ctype ( size_t __refs = 0 ) [explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

#### Parameters

*refs* Passed to the base facet class.

### 5.435.3.2 `std::ctype< wchar_t >::ctype ( __c_locale __cloc, size_t __refs = 0 ) [explicit]`

Constructor performs static initialization.

This constructor is used to construct the initial C locale facet.

#### Parameters

*cloc* Handle to C locale data.

*refs* Passed to the base facet class.

### 5.435.3.3 `virtual std::ctype< wchar_t >::~~ctype ( ) [protected, virtual]`

Destructor.

#### 5.435.4 Member Function Documentation

**5.435.4.1** `virtual bool std::ctype< wchar_t >::do_is ( mask __m, char_type __c ) const [protected, virtual]`

Test wchar\_t classification.

This function finds a mask *M* for *c* and compares it to mask *m*.

[do\\_is\(\)](#) is a hook for a derived facet to change the behavior of classifying. [do\\_is\(\)](#) must always return the same result for the same input.

##### Parameters

*c* The wchar\_t to find the mask of.

*m* The mask to compare against.

##### Returns

(*M* & *m*) != 0.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**5.435.4.2** `virtual const char_type* std::ctype< wchar_t >::do_is ( const char_type * __lo, const char_type * __hi, mask * __vec ) const [protected, virtual]`

Return a mask array.

This function finds the mask for each wchar\_t in the range [*lo*,*hi*) and successively writes it to *vec*. *vec* must have as many elements as the input.

[do\\_is\(\)](#) is a hook for a derived facet to change the behavior of classifying. [do\\_is\(\)](#) must always return the same result for the same input.

##### Parameters

*lo* Pointer to start of range.

*hi* Pointer to end of range.

*vec* Pointer to an array of mask storage.

##### Returns

*hi*.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

#### 5.435.4.3 virtual char std::ctype< wchar\_t >::do\_narrow ( char\_type, char \_\_dfault ) const [protected, virtual]

Narrow wchar\_t to char.

This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, *dfault* is returned instead. For an underived [ctype<wchar\\_t>](#) facet, *c* will be cast to char and returned.

[do\\_narrow\(\)](#) is a hook for a derived facet to change the behavior of narrowing. [do\\_narrow\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See [codecvt](#) for that.

##### Parameters

- c* The wchar\_t to convert.
- dfault* Char to return if conversion fails.

##### Returns

The converted char.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

#### 5.435.4.4 virtual const char\_type\* std::ctype< wchar\_t >::do\_narrow ( const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_dfault, char \* \_\_dest ) const [protected, virtual]

Narrow wchar\_t array to char array.

This virtual function converts each wchar\_t in the range [lo,hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any wchar\_t in the input that cannot be converted, *dfault* is used instead. For an underived [ctype<wchar\\_t>](#) facet, the argument will be copied, casting each element to char.

[do\\_narrow\(\)](#) is a hook for a derived facet to change the behavior of narrowing. [do\\_narrow\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See [codecvt](#) for that.

##### Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.

*dfault* Char to use if conversion fails.

*to* Pointer to the destination array.

### Returns

*hi*.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**5.435.4.5** `virtual const char_type* std::ctype< wchar_t >::do_scan_is ( mask  
__m, const char_type * __lo, const char_type * __hi ) const`  
`[protected, virtual]`

Find `wchar_t` matching mask.

This function searches for and returns the first `wchar_t` `c` in `[lo,hi)` for which `is(m,c)` is true.

[do\\_scan\\_is\(\)](#) is a hook for a derived facet to change the behavior of match searching.  
[do\\_is\(\)](#) must always return the same result for the same input.

### Parameters

*m* The mask to compare against.

*lo* Pointer to start of range.

*hi* Pointer to end of range.

### Returns

Pointer to a matching `wchar_t` if found, else *hi*.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**5.435.4.6** `virtual const char_type* std::ctype< wchar_t >::do_scan_not ( mask  
__m, const char_type * __lo, const char_type * __hi ) const`  
`[protected, virtual]`

Find `wchar_t` not matching mask.

This function searches for and returns a pointer to the first `wchar_t` `c` of `[lo,hi)` for which `is(m,c)` is false.

[do\\_scan\\_is\(\)](#) is a hook for a derived facet to change the behavior of match searching.  
[do\\_is\(\)](#) must always return the same result for the same input.

**Parameters**

- m* The mask to compare against.
- lo* Pointer to start of range.
- hi* Pointer to end of range.

**Returns**

Pointer to a non-matching `wchar_t` if found, else *hi*.

Implements `std::__ctype_abstract_base< wchar_t >`.

**5.435.4.7 `virtual char_type std::ctype< wchar_t >::do_tolower ( char_type )`  
`const [protected, virtual]`**

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

**Parameters**

- c* The `wchar_t` to convert.

**Returns**

The lowercase `wchar_t` if convertible, else *c*.

Implements `std::__ctype_abstract_base< wchar_t >`.

**5.435.4.8 `virtual const char_type* std::ctype< wchar_t >::do_tolower (`  
`char_type * __lo, const char_type * __hi ) const [protected,`  
`virtual]`**

Convert array to lowercase.

This virtual function converts each `wchar_t` in the range `[lo,hi)` to lowercase if possible. Other elements remain untouched.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

**Parameters**

*lo* Pointer to start of range.

*hi* Pointer to end of range.

**Returns**

*hi*.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**5.435.4.9** `virtual char_type std::ctype< wchar_t >::do_toupper ( char_type )  
const [protected, virtual]`

Convert to uppercase.

This virtual function converts the `wchar_t` argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

[do\\_toupper\(\)](#) is a hook for a derived facet to change the behavior of uppercasing. [do\\_toupper\(\)](#) must always return the same result for the same input.

**Parameters**

*c* The `wchar_t` to convert.

**Returns**

The uppercase `wchar_t` if convertible, else *c*.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**5.435.4.10** `virtual const char_type* std::ctype< wchar_t >::do_toupper ( char_type * __lo, const char_type * __hi ) const [protected, virtual]`

Convert array to uppercase.

This virtual function converts each `wchar_t` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched.

[do\\_toupper\(\)](#) is a hook for a derived facet to change the behavior of uppercasing. [do\\_toupper\(\)](#) must always return the same result for the same input.

**Parameters**

*lo* Pointer to start of range.

*hi* Pointer to end of range.

**Returns**

*hi*.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**5.435.4.11 `virtual char_type std::ctype< wchar_t >::do_widen ( char ) const [protected, virtual]`**

Widen char to wchar\_t.

This virtual function converts the char to wchar\_t using the simplest reasonable transformation. For an underived [ctype<wchar\\_t>](#) facet, the argument will be cast to wchar\_t.

[do\\_widen\(\)](#) is a hook for a derived facet to change the behavior of widening. [do\\_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

*c* The char to convert.

**Returns**

The converted wchar\_t.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**5.435.4.12 `virtual const char* std::ctype< wchar_t >::do_widen ( const char * __lo, const char * __hi, char_type * __dest ) const [protected, virtual]`**

Widen char array to wchar\_t array.

This function converts each char in the input to wchar\_t using the simplest reasonable transformation. For an underived [ctype<wchar\\_t>](#) facet, the argument will be copied, casting each element to wchar\_t.



[do\\_widen\(\)](#) is a hook for a derived facet to change the behavior of widening. [do\\_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

- lo* Pointer to start range.
- hi* Pointer to end of range.
- to* Pointer to the destination array.

#### Returns

*hi*.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**5.435.4.13** `const char_type* std::__ctype_abstract_base< wchar_t >::is ( const char_type * __lo, const char_type * __hi, mask * __vec ) const [inline, inherited]`

Return a mask array.

This function finds the mask for each `char_type` in the range `[lo,hi)` and successively writes it to `vec`. `vec` must have as many elements as the char array. It does so by returning the value of `ctype<char_type>::do_is()`.

#### Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- vec* Pointer to an array of mask storage.

#### Returns

*hi*.

Definition at line 180 of file `locale_facets.h`.

References [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >::do\\_is\(\)](#).

**5.435.4.14** `bool std::__ctype_abstract_base< wchar_t >::is ( mask __m, char_type __c ) const [inline, inherited]`

Test char\_type classification.

This function finds a mask *M* for *c* and compares it to mask *m*. It does so by returning the value of ctype<char\_type>::do\_is().

#### Parameters

- c* The char\_type to compare the mask of.
- m* The mask to compare against.

#### Returns

(*M* & *m*) != 0.

Definition at line 163 of file locale\_facets.h.

References std::\_\_ctype\_abstract\_base< \_CharT >::do\_is().

**5.435.4.15** char std::\_\_ctype\_abstract\_base< wchar\_t >::narrow ( char\_type \_\_c, char \_\_dfault ) const [inline, inherited]

Narrow char\_type to char.

This function converts the char\_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning ctype<char\_type>::do\_narrow(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

- c* The char\_type to convert.
- dfault* Char to return if conversion fails.

#### Returns

The converted char.

Definition at line 325 of file locale\_facets.h.

References std::\_\_ctype\_abstract\_base< \_CharT >::do\_narrow().

**5.435.4.16** const char\_type\* std::\_\_ctype\_abstract\_base< wchar\_t >::narrow ( const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_dfault, char \* \_\_to ) const [inline, inherited]

Narrow array to char array.

This function converts each `char_type` in the input to `char` using the simplest reasonable transformation and writes the results to the destination array. For any `char_type` in the input that cannot be converted, *dfault* is used instead. It does so by returning `ctype<char_type>::do_narrow(lo, hi, dfault, to)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- dfault* Char to use if conversion fails.
- to* Pointer to the destination array.

#### Returns

*hi*.

Definition at line 347 of file `locale_facets.h`.

References `std::__ctype_abstract_base< _CharT >::do_narrow()`.

**5.435.4.17** `const char_type* std::__ctype_abstract_base< wchar_t >::scan_is ( mask __m, const char_type * __lo, const char_type * __hi ) const [inline, inherited]`

Find `char_type` matching a mask.

This function searches for and returns the first `char_type` `c` in `[lo,hi)` for which `is(m,c)` is true. It does so by returning `ctype<char_type>::do_scan_is()`.

#### Parameters

- m* The mask to compare against.
- lo* Pointer to start of range.
- hi* Pointer to end of range.

#### Returns

Pointer to matching `char_type` if found, else *hi*.

Definition at line 196 of file `locale_facets.h`.

References `std::__ctype_abstract_base< _CharT >::do_scan_is()`.

**5.435.4.18** `const char_type* std::__ctype_abstract_base< wchar_t >::scan_not  
( mask __m, const char_type * __lo, const char_type * __hi )  
const [inline, inherited]`

Find char\_type not matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char\_type>::do\_scan\_not().

#### Parameters

*m* The mask to compare against.

*lo* Pointer to first char in range.

*hi* Pointer to end of range.

#### Returns

Pointer to non-matching char if found, else *hi*.

Definition at line 212 of file locale\_facets.h.

References std::\_\_ctype\_abstract\_base< \_CharT >::do\_scan\_not().

**5.435.4.19** `char_type std::__ctype_abstract_base< wchar_t >::tolower (   
char_type __c ) const [inline, inherited]`

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char\_type>::do\_toupper(c).

#### Parameters

*c* The char\_type to convert.

#### Returns

The lowercase char\_type if convertible, else *c*.

Definition at line 255 of file locale\_facets.h.

References std::\_\_ctype\_abstract\_base< \_CharT >::do\_tolower().

**5.435.4.20** `const char_type* std::__ctype_abstract_base< wchar_t >::tolower (char_type * __lo, const char_type * __hi ) const [inline, inherited]`

Convert array to lowercase.

This function converts each `char_type` in the range `[lo,hi)` to lowercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

#### Parameters

*lo* Pointer to start of range.

*hi* Pointer to end of range.

#### Returns

*hi*.

Definition at line 270 of file `locale_facets.h`.

References `std::__ctype_abstract_base< _CharT >::do_toupper()`.

**5.435.4.21** `char_type std::__ctype_abstract_base< wchar_t >::toupper (char_type __c ) const [inline, inherited]`

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, `'2'`), returns the argument. It does so by returning `ctype<char_type>::do_toupper(c)`.

#### Parameters

*c* The `char_type` to convert.

#### Returns

The uppercase `char_type` if convertible, else *c*.

Definition at line 226 of file `locale_facets.h`.

References `std::__ctype_abstract_base< _CharT >::do_toupper()`.

**5.435.4.22** `const char_type* std::__ctype_abstract_base< wchar_t >::toupper ( char_type * __lo, const char_type * __hi ) const [inline, inherited]`

Convert array to uppercase.

This function converts each `char_type` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

#### Parameters

*lo* Pointer to start of range.

*hi* Pointer to end of range.

#### Returns

*hi*.

Definition at line 241 of file `locale_facets.h`.

References `std::__ctype_abstract_base< _CharT >::do_toupper()`.

**5.435.4.23** `char_type std::__ctype_abstract_base< wchar_t >::widen ( char __c ) const [inline, inherited]`

Widen `char` to `char_type`.

This function converts the `char` argument to `char_type` using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

*c* The `char` to convert.

#### Returns

The converted `char_type`.

Definition at line 287 of file `locale_facets.h`.

References `std::__ctype_abstract_base< _CharT >::do_widen()`.

**5.435.4.24** `const char* std::__ctype_abstract_base< wchar_t >::widen (`  
`const char * __lo, const char * __hi, char_type * __to ) const`  
`[inline, inherited]`

Widen array to char\_type.

This function converts each char in the input to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- to* Pointer to the destination array.

#### Returns

*hi*.

Definition at line 306 of file locale\_facets.h.

References std::\_\_ctype\_abstract\_base< \_CharT >::do\_widen().

### 5.435.5 Member Data Documentation

**5.435.5.1** `locale::id std::ctype< wchar_t >::id [static]`

The facet id for [ctype<wchar\\_t>](#)

Definition at line 1199 of file locale\_facets.h.

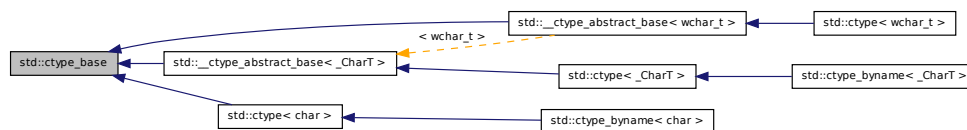
The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

## 5.436 std::ctype\_base Struct Reference

Base class for ctype.

Inheritance diagram for std::ctype\_base:



### Public Types

- typedef const int \* **\_\_to\_type**
- typedef unsigned short **mask**

### Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

#### 5.436.1 Detailed Description

Base class for ctype.

Definition at line 42 of file ctype\_base.h.

The documentation for this struct was generated from the following file:

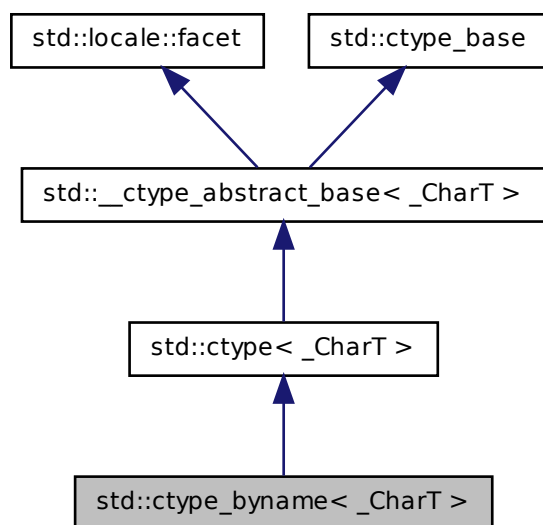
- [ctype\\_base.h](#)



## 5.437 std::ctype\_byname< \_CharT > Class Template Reference

class [ctype\\_byname](#) [22.2.1.2].

Inheritance diagram for std::ctype\_byname< \_CharT >:



### Public Types

- typedef const int \* [\\_\\_to\\_type](#)
- typedef `_CharT` [char\\_type](#)
- typedef [ctype](#)< `_CharT` >::mask **mask**

### Public Member Functions

- **ctype\_byname** (const char \* \_\_s, size\_t \_\_refs=0)
- bool **is** (mask \_\_m, [char\\_type](#) \_\_c) const
- const [char\\_type](#) \* **is** (const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi, mask \* \_\_vec) const
- const [char\\_type](#) \* **narrow** (const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi, char \_\_dfault, char \* \_\_to) const

- char [narrow](#) (char\_type \_\_c, char \_\_dfault) const
- const char\_type \* [scan\\_is](#) (mask \_\_m, const char\_type \*\_\_lo, const char\_type \*\_\_hi) const
- const char\_type \* [scan\\_not](#) (mask \_\_m, const char\_type \*\_\_lo, const char\_type \*\_\_hi) const
- char\_type [tolower](#) (char\_type \_\_c) const
- const char\_type \* [tolower](#) (char\_type \*\_\_lo, const char\_type \*\_\_hi) const
- const char\_type \* [toupper](#) (char\_type \*\_\_lo, const char\_type \*\_\_hi) const
- char\_type [toupper](#) (char\_type \_\_c) const
- const char \* [widen](#) (const char \*\_\_lo, const char \*\_\_hi, char\_type \*\_\_to) const
- char\_type [widen](#) (char \_\_c) const

### Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static [locale::id](#) **id**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

### Protected Member Functions

- virtual bool [do\\_is](#) (mask \_\_m, char\_type \_\_c) const
- virtual const char\_type \* [do\\_is](#) (const char\_type \*\_\_lo, const char\_type \*\_\_hi, mask \*\_\_vec) const
- virtual const char\_type \* [do\\_narrow](#) (const char\_type \*\_\_lo, const char\_type \*\_\_hi, char \_\_dfault, char \*\_\_dest) const
- virtual char [do\\_narrow](#) (char\_type \_\_c, char \_\_dfault) const
- virtual const char\_type \* [do\\_scan\\_is](#) (mask \_\_m, const char\_type \*\_\_lo, const char\_type \*\_\_hi) const
- virtual const char\_type \* [do\\_scan\\_not](#) (mask \_\_m, const char\_type \*\_\_lo, const char\_type \*\_\_hi) const
- virtual char\_type [do\\_tolower](#) (char\_type \_\_c) const
- virtual const char\_type \* [do\\_tolower](#) (char\_type \*\_\_lo, const char\_type \*\_\_hi) const

- virtual const [char\\_type](#) \* [do\\_toupper](#) ([char\\_type](#) \*\_\_lo, const [char\\_type](#) \*\_\_hi) const
- virtual [char\\_type](#) [do\\_toupper](#) ([char\\_type](#) \_\_c) const
- virtual const char \* [do\\_widen](#) (const char \*\_\_lo, const char \*\_\_hi, [char\\_type](#) \*\_\_dest) const
- virtual [char\\_type](#) [do\\_widen](#) (char \_\_c) const

### Static Protected Member Functions

- static \_\_c\_locale [\\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale [\\_S\\_get\\_c\\_locale](#) ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static \_\_c\_locale [\\_S\\_lc\\_ctype\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \*\_\_s)

### Friends

- class [locale::Impl](#)

#### 5.437.1 Detailed Description

**template<typename \_CharT> class std::ctype\_byname<\_CharT>**

class [ctype\\_byname](#) [22.2.1.2].

Definition at line 1468 of file locale\_facets.h.

#### 5.437.2 Member Typedef Documentation

**5.437.2.1** **template<typename \_CharT> typedef \_CharT std::ctype<\_CharT>::char\_type [inherited]**

Typedef for the template parameter.

Reimplemented from [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Definition at line 610 of file locale\_facets.h.

### 5.437.3 Member Function Documentation

**5.437.3.1** `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_is ( const char_type * __lo, const char_type * __hi, mask * __vec ) const` `[protected, virtual, inherited]`

Return a mask array.

This function finds the mask for each `char_type` in the range `[lo,hi)` and successively writes it to `vec`. `vec` must have as many elements as the input.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

#### Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- vec* Pointer to an array of mask storage.

#### Returns

*hi*.

Implements `std::__ctype_abstract_base<_CharT>`.

**5.437.3.2** `template<typename _CharT> virtual bool std::ctype<_CharT>::do_is ( mask __m, char_type __c ) const` `[protected, virtual, inherited]`

Test `char_type` classification.

This function finds a mask `M` for `c` and compares it to mask `m`.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

#### Parameters

- c* The `char_type` to find the mask of.
- m* The mask to compare against.

#### Returns

$(M \& m) \neq 0$ .

Implements `std::__ctype_abstract_base<_CharT>`.

**5.437.3.3** `template<typename _CharT> virtual char std::ctype< _CharT  
>::do_narrow( char_type, char __dfault ) const [protected,  
virtual, inherited]`

Narrow `char_type` to `char`.

This virtual function converts the argument to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

- c* The `char_type` to convert.
- dfault* Char to return if conversion fails.

#### Returns

The converted `char`.

Implements `std::__ctype_abstract_base< _CharT >`.

Referenced by `std::ctype< char >::narrow()`.

**5.437.3.4** `template<typename _CharT> virtual const char_type* std::ctype<  
_CharT >::do_narrow( const char_type * __lo, const char_type *  
__hi, char __dfault, char * __dest ) const [protected,  
virtual, inherited]`

Narrow `char_type` array to `char`.

This virtual function converts each `char_type` in the range `[lo,hi)` to `char` using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, `dfault` is used instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

- lo* Pointer to start of range.

*hi* Pointer to end of range.

*dfault* Char to use if conversion fails.

*to* Pointer to the destination array.

### Returns

*hi*.

Implements [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >](#).

**5.437.3.5** `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_scan_is ( mask __m, const char_type * __lo, const char_type * __hi ) const [protected, virtual, inherited]`

Find char\_type matching mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is true.

[do\\_scan\\_is\(\)](#) is a hook for a derived facet to change the behavior of match searching. [do\\_is\(\)](#) must always return the same result for the same input.

### Parameters

*m* The mask to compare against.

*lo* Pointer to start of range.

*hi* Pointer to end of range.

### Returns

Pointer to a matching char\_type if found, else *hi*.

Implements [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >](#).

**5.437.3.6** `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_scan_not ( mask __m, const char_type * __lo, const char_type * __hi ) const [protected, virtual, inherited]`

Find char\_type not matching mask.

This function searches for and returns a pointer to the first `char_type` `c` of `[lo,hi)` for which `is(m,c)` is false.

`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.

#### Parameters

- m* The mask to compare against.
- lo* Pointer to start of range.
- hi* Pointer to end of range.

#### Returns

Pointer to a non-matching `char_type` if found, else *hi*.

Implements `std::__ctype_abstract_base<_CharT>`.

**5.437.3.7** `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_tolower ( char_type * __lo, const char_type * __hi ) const [protected, virtual, inherited]`

Convert array to lowercase.

This virtual function converts each `char_type` in the range `[lo,hi)` to lowercase if possible. Other elements remain untouched.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

#### Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.

#### Returns

*hi*.

Implements `std::__ctype_abstract_base<_CharT>`.

**5.437.3.8** `template<typename _CharT> virtual char_type std::ctype<_CharT>::do_tolower ( char_type ) const [protected, virtual, inherited]`

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

[do\\_tolower\(\)](#) is a hook for a derived facet to change the behavior of lowercasing. [do\\_tolower\(\)](#) must always return the same result for the same input.

#### Parameters

*c* The char\_type to convert.

#### Returns

The lowercase char\_type if convertible, else *c*.

Implements [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >](#).

Referenced by [std::ctype< char >::tolower\(\)](#).

**5.437.3.9** `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_toupper ( char_type * __lo, const char_type * __hi ) const [protected, virtual, inherited]`

Convert array to uppercase.

This virtual function converts each char\_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched.

[do\\_toupper\(\)](#) is a hook for a derived facet to change the behavior of uppercasing. [do\\_toupper\(\)](#) must always return the same result for the same input.

#### Parameters

*lo* Pointer to start of range.

*hi* Pointer to end of range.

#### Returns

*hi*.

Implements [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >](#).

**5.437.3.10** `template<typename _CharT> virtual char_type std::ctype<_CharT>::do_toupper ( char_type ) const [protected, virtual, inherited]`



Convert to uppercase.

This virtual function converts the `char_type` argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

[do\\_toupper\(\)](#) is a hook for a derived facet to change the behavior of uppercasing. [do\\_toupper\(\)](#) must always return the same result for the same input.

#### Parameters

*c* The `char_type` to convert.

#### Returns

The uppercase `char_type` if convertible, else *c*.

Implements [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >](#).

Referenced by `std::ctype< char >::toupper()`.

**5.437.3.11** `template<typename _CharT> virtual char_type std::ctype<  
_CharT >::do_widen( char ) const [protected, virtual,  
inherited]`

Widen char.

This virtual function converts the `char` to `char_type` using the simplest reasonable transformation.

[do\\_widen\(\)](#) is a hook for a derived facet to change the behavior of widening. [do\\_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

*c* The `char` to convert.

#### Returns

The converted `char_type`

Implements [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >](#).

Referenced by `std::ctype< char >::widen()`.

**5.437.3.12** `template<typename _CharT> virtual const char* std::ctype<_CharT>::do_widen ( const char * __lo, const char * __hi, char_type * __dest ) const [protected, virtual, inherited]`

Widen char array.

This function converts each char in the input to char\_type using the simplest reasonable transformation.

[do\\_widen\(\)](#) is a hook for a derived facet to change the behavior of widening. [do\\_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

- lo* Pointer to start range.
- hi* Pointer to end of range.
- to* Pointer to the destination array.

#### Returns

*hi*.

Implements [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >](#).

**5.437.3.13** `template<typename _CharT> bool std::__ctype_abstract_base<_CharT>::is ( mask __m, char_type __c ) const [inline, inherited]`

Test char\_type classification.

This function finds a mask *M* for *c* and compares it to mask *m*. It does so by returning the value of [ctype<char\\_type>::do\\_is\(\)](#).

#### Parameters

- c* The char\_type to compare the mask of.
- m* The mask to compare against.

#### Returns

(*M* & *m*) != 0.

Definition at line 163 of file locale\_facets.h.

Referenced by std::regex\_traits<\_Ch\_type>::isctype(), and std::basic\_istream<\_CharT, \_Traits>::sentry::sentry().

**5.437.3.14** `template<typename _CharT> const char_type*  
std::__ctype_abstract_base<_CharT>::is ( const char_type *  
__lo, const char_type * __hi, mask * __vec ) const [inline,  
inherited]`

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of `ctype<char_type>::do_is()`.

#### Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- vec* Pointer to an array of mask storage.

#### Returns

*hi*.

Definition at line 180 of file locale\_facets.h.

**5.437.3.15** `template<typename _CharT> char std::__ctype_abstract_base<  
_CharT>::narrow ( char_type __c, char __dfault ) const  
[inline, inherited]`

Narrow char\_type to char.

This function converts the char\_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning `ctype<char_type>::do_narrow(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

- c* The char\_type to convert.
- dfault* Char to return if conversion fails.

**Returns**

The converted char.

Definition at line 325 of file locale\_facets.h.

Referenced by std::time\_put< \_CharT, \_OutIter >::put().

**5.437.3.16** `template<typename _CharT> const char_type*  
std::__ctype_abstract_base< _CharT >::narrow ( const char_type  
* __lo, const char_type * __hi, char __dfault, char * __to ) const  
[inline, inherited]`

Narrow array to char array.

This function converts each `char_type` in the input to `char` using the simplest reasonable transformation and writes the results to the destination array. For any `char_type` in the input that cannot be converted, *dfault* is used instead. It does so by returning `ctype<char_type>::do_narrow(lo, hi, dfault, to)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

*lo* Pointer to start of range.

*hi* Pointer to end of range.

*dfault* Char to use if conversion fails.

*to* Pointer to the destination array.

**Returns**

*hi*.

Definition at line 347 of file locale\_facets.h.

**5.437.3.17** `template<typename _CharT> const char_type*  
std::__ctype_abstract_base< _CharT >::scan_is ( mask __m,  
const char_type * __lo, const char_type * __hi ) const [inline,  
inherited]`

Find `char_type` matching a mask.

This function searches for and returns the first `char_type` `c` in `[lo,hi)` for which `is(m,c)` is true. It does so by returning `ctype<char_type>::do_scan_is()`.

**Parameters**

*m* The mask to compare against.

*lo* Pointer to start of range.

*hi* Pointer to end of range.

**Returns**

Pointer to matching char\_type if found, else *hi*.

Definition at line 196 of file locale\_facets.h.

```
5.437.3.18 template<typename _CharT> const char_type*
std::__ctype_abstract_base<_CharT>::scan_not (mask __m,
const char_type * __lo, const char_type * __hi) const [inline,
inherited]
```

Find char\_type not matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is false. It does so by returning [ctype<char\\_type>::do\\_scan\\_not\(\)](#).

**Parameters**

*m* The mask to compare against.

*lo* Pointer to first char in range.

*hi* Pointer to end of range.

**Returns**

Pointer to non-matching char if found, else *hi*.

Definition at line 212 of file locale\_facets.h.

```
5.437.3.19 template<typename _CharT> const char_type*
std::__ctype_abstract_base<_CharT>::tolower (char_type *
__lo, const char_type * __hi) const [inline, inherited]
```

Convert array to lowercase.

This function converts each char\_type in the range [lo,hi) to lowercase if possible. Other elements remain untouched. It does so by returning [ctype<char\\_type>::do\\_toupper\(lo, hi\)](#).

**Parameters**

*lo* Pointer to start of range.

*hi* Pointer to end of range.

**Returns**

*hi*.

Definition at line 270 of file locale\_facets.h.

**5.437.3.20** `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::tolower ( char_type __c ) const [inline, inherited]`

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char\_type>::do\_toupper(c).

**Parameters**

*c* The char\_type to convert.

**Returns**

The lowercase char\_type if convertible, else *c*.

Definition at line 255 of file locale\_facets.h.

**5.437.3.21** `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::toupper ( char_type * __lo, const char_type * __hi ) const [inline, inherited]`

Convert array to uppercase.

This function converts each char\_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning ctype<char\_type>::do\_toupper(lo, hi).

**Parameters**

*lo* Pointer to start of range.

*hi* Pointer to end of range.

#### Returns

*hi*.

Definition at line 241 of file locale\_facets.h.

**5.437.3.22** `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::toupper ( char_type __c ) const [inline, inherited]`

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_toupper()`.

#### Parameters

*c* The char\_type to convert.

#### Returns

The uppercase char\_type if convertible, else *c*.

Definition at line 226 of file locale\_facets.h.

**5.437.3.23** `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::widen ( char __c ) const [inline, inherited]`

Widen char to char\_type.

This function converts the char argument to char\_type using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

*c* The char to convert.

#### Returns

The converted char\_type.

Definition at line 287 of file locale\_facets.h.

Referenced by std::money\_get< \_CharT, \_InIter >::do\_get(), std::time\_put< \_CharT, \_OutIter >::do\_put(), std::money\_put< \_CharT, \_OutIter >::do\_put(), std::regex\_traits< \_Ch\_type >::isctype(), and std::operator<<().

**5.437.3.24** `template<typename _CharT> const char* std::__ctype_abstract_base< _CharT >::widen ( const char * __lo, const char * __hi, char_type * __to ) const [inline, inherited]`

Widen array to char\_type.

This function converts each char in the input to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- to* Pointer to the destination array.

#### Returns

*hi*.

Definition at line 306 of file locale\_facets.h.

### 5.437.4 Member Data Documentation

**5.437.4.1** `template<typename _CharT> locale::id std::ctype< _CharT >::id [static, inherited]`

The facet id for ctype<char\_type>

Definition at line 614 of file locale\_facets.h.

The documentation for this class was generated from the following file:

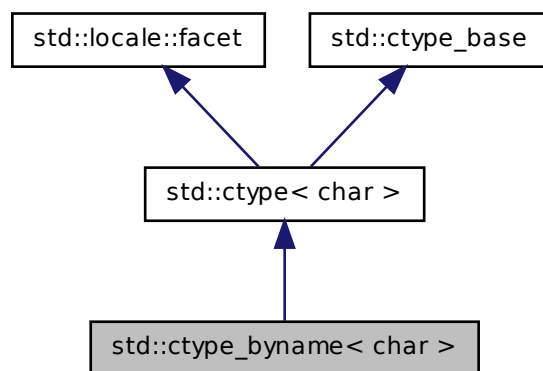
- [locale\\_facets.h](#)



**5.438 `std::ctype_byname< char >` Class Template Reference**

22.2.1.4 Class `ctype_byname` specializations.

Inheritance diagram for `std::ctype_byname< char >`:

**Public Types**

- typedef const int \* `__to_type`
- typedef char `char_type`
- typedef unsigned short `mask`

**Public Member Functions**

- `ctype_byname` (const char \* \_\_s, size\_t \_\_refs=0)
- bool `is` (mask \_\_m, char \_\_c) const
- const char \* `is` (const char \* \_\_lo, const char \* \_\_hi, mask \* \_\_vec) const
- const `char_type` \* `narrow` (const `char_type` \* \_\_lo, const `char_type` \* \_\_hi, char \_\_dfault, char \* \_\_to) const
- char `narrow` (`char_type` \_\_c, char \_\_dfault) const
- const char \* `scan_is` (mask \_\_m, const char \* \_\_lo, const char \* \_\_hi) const
- const char \* `scan_not` (mask \_\_m, const char \* \_\_lo, const char \* \_\_hi) const
- const mask \* `table` () const throw ()
- const `char_type` \* `tolower` (`char_type` \* \_\_lo, const `char_type` \* \_\_hi) const

- `char_type tolower (char_type __c) const`
- `char_type toupper (char_type __c) const`
- `const char_type * toupper (char_type *__lo, const char_type *__hi) const`
- `const char * widen (const char *__lo, const char *__hi, char_type *__to) const`
- `char_type widen (char __c) const`

### Static Public Member Functions

- `static const mask * classic_table () throw ()`

### Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static `locale::id id`
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const `size_t table_size`
- static const mask **upper**
- static const mask **xdigit**

### Protected Member Functions

- virtual `char do_narrow (char_type __c, char) const`
- virtual const `char_type * do_narrow (const char_type *__lo, const char_type *__hi, char, char *__dest) const`
- virtual `char_type do_tolower (char_type) const`
- virtual const `char_type * do_tolower (char_type *__lo, const char_type *__hi) const`
- virtual `char_type do_toupper (char_type) const`
- virtual const `char_type * do_toupper (char_type *__lo, const char_type *__hi) const`
- virtual `char_type do_widen (char __c) const`
- virtual const `char * do_widen (const char *__lo, const char *__hi, char_type *__dest) const`

**Static Protected Member Functions**

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static const char \* `_S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

**Protected Attributes**

- `__c_locale _M_c_locale_ctype`
- bool `_M_del`
- char `_M_narrow` [1+static\_cast< unsigned char >(-1)]
- char `_M_narrow_ok`
- const mask \* `_M_table`
- `__to_type _M_tolower`
- `__to_type _M_toupper`
- char `_M_widen` [1+static\_cast< unsigned char >(-1)]
- char `_M_widen_ok`

**Friends**

- class `locale::_Impl`

**5.438.1 Detailed Description**

`template<> class std::ctype_byname< char >`

22.2.1.4 Class [ctype\\_byname](#) specializations.

Definition at line 1483 of file `locale_facets.h`.

**5.438.2 Member Typedef Documentation****5.438.2.1 `typedef char std::ctype< char >::char_type` [inherited]**

Typedef for the template parameter `char`.

Definition at line 680 of file `locale_facets.h`.

### 5.438.3 Member Function Documentation

**5.438.3.1** `static const mask* std::ctype< char >::classic_table ( ) throw ()`  
`[static, inherited]`

Returns a pointer to the C locale mask table.

**5.438.3.2** `virtual char std::ctype< char >::do_narrow ( char_type __c, char`  
`) const [inline, protected, virtual, inherited]`

Narrow char.

This virtual function converts the char to char using the simplest reasonable transformation. If the conversion fails, *dfault* is returned instead. For an underived `ctype<char>` facet, *c* will be returned unchanged.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

*c* The char to convert.

*dfault* Char to return if conversion fails.

#### Returns

The converted char.

Definition at line 1125 of file `locale_facets.h`.

**5.438.3.3** `virtual const char_type* std::ctype< char >::do_narrow ( const`  
`char_type * __lo, const char_type * __hi, char, char * __dest )`  
`const [inline, protected, virtual, inherited]`

Narrow char array to char array.

This virtual function converts each char in the range [lo,hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *dfault* is used instead. For an underived `ctype<char>` facet, the argument will be copied unchanged.

[do\\_narrow\(\)](#) is a hook for a derived facet to change the behavior of narrowing. [do\\_narrow\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- dfault* Char to use if conversion fails.
- to* Pointer to the destination array.

#### Returns

*hi*.

Definition at line 1151 of file `locale_facets.h`.

**5.438.3.4** `virtual const char_type* std::ctype< char >::do_tolower ( char_type * __lo, const char_type * __hi ) const` `[protected, virtual, inherited]`

Convert array to lowercase.

This virtual function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

[do\\_tolower\(\)](#) is a hook for a derived facet to change the behavior of lowercasing. [do\\_tolower\(\)](#) must always return the same result for the same input.

#### Parameters

- lo* Pointer to first char in range.
- hi* Pointer to end of range.

#### Returns

*hi*.

**5.438.3.5** `virtual char_type std::ctype< char >::do_tolower ( char_type ) const` `[protected, virtual, inherited]`

Convert to lowercase.

This virtual function converts the `char` argument to lowercase if possible. If not possible (for example, `'2'`), returns the argument.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

#### Parameters

*c* The `char` to convert.

#### Returns

The lowercase `char` if convertible, else *c*.

**5.438.3.6** `virtual const char_type* std::ctype< char >::do_toupper ( char_type * __lo, const char_type * __hi ) const` `[protected, virtual, inherited]`

Convert array to uppercase.

This virtual function converts each `char` in the range `[lo,hi)` to uppercase if possible. Other `chars` remain untouched.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

#### Parameters

*lo* Pointer to start of range.

*hi* Pointer to end of range.

#### Returns

*hi*.

**5.438.3.7** `virtual char_type std::ctype< char >::do_toupper ( char_type ) const` `[protected, virtual, inherited]`

Convert to uppercase.

This virtual function converts the `char` argument to uppercase if possible. If not possible (for example, `'2'`), returns the argument.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

#### Parameters

*c* The char to convert.

#### Returns

The uppercase char if convertible, else *c*.

#### 5.438.3.8 virtual char\_type std::ctype< char >::do\_widen ( char \_\_c ) const [inline, protected, virtual, inherited]

Widen char.

This virtual function converts the char to char using the simplest reasonable transformation. For an underived `ctype<char>` facet, the argument will be returned unchanged.

`do_widen()` is a hook for a derived facet to change the behavior of widening. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

*c* The char to convert.

#### Returns

The converted character.

Definition at line 1076 of file `locale_facets.h`.

#### 5.438.3.9 virtual const char\* std::ctype< char >::do\_widen ( const char \* \_\_lo, const char \* \_\_hi, char\_type \* \_\_dest ) const [inline, protected, virtual, inherited]

Widen char array.

This function converts each char in the range [lo,hi) to char using the simplest reasonable transformation. For an underived `ctype<char>` facet, the argument will be copied unchanged.

`do_widen()` is a hook for a derived facet to change the behavior of widening. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- to* Pointer to the destination array.

#### Returns

*hi*.

Definition at line 1099 of file locale\_facets.h.

**5.438.3.10** `const char * std::ctype< char >::is ( const char * __lo, const char * __hi, mask * __vec ) const [inline, inherited]`

Return a mask array.

This function finds the mask for each char in the range [lo, hi) and successively writes it to vec. vec must have as many elements as the char array.

#### Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- vec* Pointer to an array of mask storage.

#### Returns

*hi*.

Definition at line 48 of file ctype\_inline.h.

**5.438.3.11** `bool std::ctype< char >::is ( mask __m, char __c ) const [inline, inherited]`

Test char classification.

This function compares the mask table[c] to *m*.

#### Parameters

- c* The char to compare the mask of.



*m* The mask to compare against.

### Returns

True if *m* & table[*c*] is true, false otherwise.

Definition at line 43 of file ctype\_inline.h.

**5.438.3.12** `char std::ctype< char >::narrow ( char_type __c, char __dfault )  
const [inline, inherited]`

Narrow char.

This function converts the char to char using the simplest reasonable transformation. If the conversion fails, *dfault* is returned instead. For an underived `ctype<char>` facet, *c* will be returned unchanged.

This function works as if it returns `ctype<char>::do_narrow(c)`. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

### Parameters

*c* The char to convert.

*dfault* Char to return if conversion fails.

### Returns

The converted character.

Definition at line 924 of file locale\_facets.h.

References `std::ctype< _CharT >::do_narrow()`.

**5.438.3.13** `const char_type* std::ctype< char >::narrow ( const char_type *  
__lo, const char_type * __hi, char __dfault, char * __to ) const  
[inline, inherited]`

Narrow char array.

This function converts each char in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *dfault* is used instead. For an underived `ctype<char>` facet, the argument will be copied unchanged.

This function works as if it returns `ctype<char>::do_narrow(lo, hi, dfault, to)`. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- dfault* Char to use if conversion fails.
- to* Pointer to the destination array.

#### Returns

*hi*.

Definition at line 957 of file `locale_facets.h`.

References `std::ctype<_CharT>::do_narrow()`.

**5.438.3.14** `const char * std::ctype< char >::scan_is ( mask __m, const char * __lo, const char * __hi ) const [inline, inherited]`

Find char matching a mask.

This function searches for and returns the first char in `[lo,hi)` for which `is(m,char)` is true.

#### Parameters

- m* The mask to compare against.
- lo* Pointer to start of range.
- hi* Pointer to end of range.

#### Returns

Pointer to a matching char if found, else *hi*.

Definition at line 57 of file `ctype_inline.h`.

**5.438.3.15** `const char * std::ctype< char >::scan_not ( mask __m, const char * __lo, const char * __hi ) const [inline, inherited]`

Find char not matching a mask.

This function searches for and returns a pointer to the first char in [lo,hi) for which `is(m,char)` is false.

#### Parameters

- m* The mask to compare against.
- lo* Pointer to start of range.
- hi* Pointer to end of range.

#### Returns

Pointer to a non-matching char if found, else *hi*.

Definition at line 67 of file `ctype_inline.h`.

**5.438.3.16** `const mask* std::ctype< char >::table ( ) const throw ()`  
[inline, inherited]

Returns a pointer to the mask table provided to the constructor, or /// the default from [classic\\_table\(\)](#) if none was provided.

Definition at line 975 of file `locale_facets.h`.

**5.438.3.17** `char_type std::ctype< char >::tolower ( char_type __c ) const`  
[inline, inherited]

Convert to lowercase.

This function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

[tolower\(\)](#) acts as if it returns `ctype<char>::do_tolower(c)`. [do\\_tolower\(\)](#) must always return the same result for the same input.

#### Parameters

- c* The char to convert.

#### Returns

The lowercase char if convertible, else *c*.

Definition at line 829 of file `locale_facets.h`.

References `std::ctype< _CharT >::do_tolower()`.

**5.438.3.18** `const char_type* std::ctype< char >::tolower ( char_type * __lo,  
const char_type * __hi ) const [inline, inherited]`

Convert array to lowercase.

This function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

`tolower()` acts as if it returns `ctype<char>::do_tolower(lo, hi)`. `do_tolower()` must always return the same result for the same input.

#### Parameters

*lo* Pointer to first char in range.

*hi* Pointer to end of range.

#### Returns

*hi*.

Definition at line 846 of file locale\_facets.h.

References `std::ctype< _CharT >::do_tolower()`.

**5.438.3.19** `const char_type* std::ctype< char >::toupper ( char_type * __lo,  
const char_type * __hi ) const [inline, inherited]`

Convert array to uppercase.

This function converts each char in the range [lo,hi) to uppercase if possible. Other chars remain untouched.

`toupper()` acts as if it returns `ctype<char>::do_toupper(lo, hi)`. `do_toupper()` must always return the same result for the same input.

#### Parameters

*lo* Pointer to first char in range.

*hi* Pointer to end of range.

#### Returns

*hi*.

Definition at line 813 of file locale\_facets.h.

References `std::ctype< _CharT >::do_toupper()`.

**5.438.3.20 char\_type std::ctype< char >::toupper ( char\_type \_\_c ) const  
[inline, inherited]**

Convert to uppercase.

This function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

[toupper\(\)](#) acts as if it returns ctype<char>::do\_toupper(c). [do\\_toupper\(\)](#) must always return the same result for the same input.

**Parameters**

*c* The char to convert.

**Returns**

The uppercase char if convertible, else *c*.

Definition at line 796 of file locale\_facets.h.

References std::ctype< \_CharT >::do\_toupper().

**5.438.3.21 char\_type std::ctype< char >::widen ( char \_\_c ) const  
[inline, inherited]**

Widen char.

This function converts the char to char\_type using the simplest reasonable transformation. For an underived [ctype<char>](#) facet, the argument will be returned unchanged.

This function works as if it returns ctype<char>::do\_widen(c). [do\\_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

*c* The char to convert.

**Returns**

The converted character.

Definition at line 866 of file locale\_facets.h.

References std::ctype< \_CharT >::do\_widen().

**5.438.3.22** `const char* std::ctype< char >::widen ( const char * __lo, const char * __hi, char_type * __to ) const [inline, inherited]`

Widen char array.

This function converts each char in the input to char using the simplest reasonable transformation. For an underived `ctype<char>` facet, the argument will be copied unchanged.

This function works as if it returns `ctype<char>::do_widen(c)`. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

- lo* Pointer to first char in range.
- hi* Pointer to end of range.
- to* Pointer to the destination array.

#### Returns

*hi*.

Definition at line 893 of file `locale_facets.h`.

References `std::ctype< _CharT >::do_widen()`.

### 5.438.4 Member Data Documentation

**5.438.4.1** `locale::id std::ctype< char >::id [static, inherited]`

The facet id for `ctype<char>`

Definition at line 697 of file `locale_facets.h`.

**5.438.4.2** `const size_t std::ctype< char >::table_size [static, inherited]`

The size of the mask table. It is `SCHAR_MAX + 1`.

Definition at line 699 of file `locale_facets.h`.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

## 5.439 `std::decay< _Tp >` Class Template Reference

`decay`

### Public Types

- `typedef __decay_selector< __remove_type >::__type type`

### 5.439.1 Detailed Description

`template<typename _Tp> class std::decay< _Tp >`

`decay`

Definition at line 880 of file `type_traits`.

The documentation for this class was generated from the following file:

- [type\\_traits](#)

## 5.440 `std::decimal::decimal128` Class Reference

3.2.4 Class [decimal128](#).

### Public Types

- `typedef float __decfloat128 __attribute__((mode(TD)))`

### Public Member Functions

- `decimal128` ([decimal32](#) d32)
- `decimal128` (float \_\_r)
- `decimal128` (unsigned int \_\_z)
- `decimal128` (long \_\_z)
- `decimal128` (double \_\_r)
- `decimal128` (unsigned long \_\_z)
- `decimal128` (long long \_\_z)
- `decimal128` ([decimal64](#) d64)
- `decimal128` (long double \_\_r)
- `decimal128` (unsigned long long \_\_z)

- [decimal128](#) (\_\_decfloat128 \_\_z)
- **decimal128** (int \_\_z)
- \_\_decfloat128 **getval** (void)
- void **setval** (\_\_decfloat128 \_\_x)
- [decimal128](#) & **operator\*=** ([decimal32](#) \_\_rhs)
- [decimal128](#) & **operator\*=** ([decimal64](#) \_\_rhs)
- [decimal128](#) & **operator\*=** ([decimal128](#) \_\_rhs)
- [decimal128](#) & **operator\*=** (int \_\_rhs)
- [decimal128](#) & **operator\*=** (unsigned int \_\_rhs)
- [decimal128](#) & **operator\*=** (long \_\_rhs)
- [decimal128](#) & **operator\*=** (unsigned long \_\_rhs)
- [decimal128](#) & **operator\*=** (unsigned long long \_\_rhs)
- [decimal128](#) & **operator\*=** (long long \_\_rhs)
- [decimal128](#) & **operator++** ()
- [decimal128](#) **operator++** (int)
- [decimal128](#) & **operator+=** (int \_\_rhs)
- [decimal128](#) & **operator+=** ([decimal32](#) \_\_rhs)
- [decimal128](#) & **operator+=** ([decimal64](#) \_\_rhs)
- [decimal128](#) & **operator+=** ([decimal128](#) \_\_rhs)
- [decimal128](#) & **operator+=** (unsigned int \_\_rhs)
- [decimal128](#) & **operator+=** (long \_\_rhs)
- [decimal128](#) & **operator+=** (unsigned long \_\_rhs)
- [decimal128](#) & **operator+=** (long long \_\_rhs)
- [decimal128](#) & **operator+=** (unsigned long long \_\_rhs)
- [decimal128](#) & **operator--** ()
- [decimal128](#) **operator--** (int)
- [decimal128](#) & **operator-=** (long long \_\_rhs)
- [decimal128](#) & **operator-=** (long \_\_rhs)
- [decimal128](#) & **operator-=** (int \_\_rhs)
- [decimal128](#) & **operator-=** ([decimal32](#) \_\_rhs)
- [decimal128](#) & **operator-=** (unsigned long \_\_rhs)
- [decimal128](#) & **operator-=** ([decimal128](#) \_\_rhs)
- [decimal128](#) & **operator-=** (unsigned long long \_\_rhs)
- [decimal128](#) & **operator-=** (unsigned int \_\_rhs)
- [decimal128](#) & **operator-=** ([decimal64](#) \_\_rhs)
- [decimal128](#) & **operator/=** ([decimal64](#) \_\_rhs)
- [decimal128](#) & **operator/=** (long \_\_rhs)
- [decimal128](#) & **operator/=** (long long \_\_rhs)
- [decimal128](#) & **operator/=** ([decimal32](#) \_\_rhs)
- [decimal128](#) & **operator/=** (unsigned int \_\_rhs)
- [decimal128](#) & **operator/=** (unsigned long \_\_rhs)
- [decimal128](#) & **operator/=** (unsigned long long \_\_rhs)
- [decimal128](#) & **operator/=** ([decimal128](#) \_\_rhs)
- [decimal128](#) & **operator/=** (int \_\_rhs)



### 5.440.1 Detailed Description

3.2.4 Class [decimal128](#).

Definition at line 393 of file `decimal`.

### 5.440.2 Constructor & Destructor Documentation

#### 5.440.2.1 `std::decimal::decimal128::decimal128 ( __decfloat128 __z ) [inline]`

Conforming extension: Conversion from scalar decimal type.

Definition at line 418 of file `decimal`.

The documentation for this class was generated from the following file:

- [decimal](#)

## 5.441 `std::decimal::decimal32` Class Reference

3.2.2 Class [decimal32](#).

### Public Types

- typedef float `__decfloat32 __attribute__((mode(SD)))`

### Public Member Functions

- **decimal32** ([decimal64](#) \_\_d64)
- **decimal32** (float \_\_r)
- **decimal32** (unsigned int \_\_z)
- **decimal32** (long \_\_z)
- **decimal32** (double \_\_r)
- **decimal32** (unsigned long \_\_z)
- **decimal32** (long long \_\_z)
- **decimal32** ([decimal128](#) \_\_d128)
- **decimal32** (long double \_\_r)
- **decimal32** (unsigned long long \_\_z)
- [decimal32](#) (\_\_decfloat32 \_\_z)
- **decimal32** (int \_\_z)
- `__decfloat32 __getval` (void)

- void **\_\_setval** (\_\_decfloat32 \_\_x)
- decimal32 & **operator\*=** (decimal32 \_\_rhs)
- decimal32 & **operator\*=** (decimal64 \_\_rhs)
- decimal32 & **operator\*=** (decimal128 \_\_rhs)
- decimal32 & **operator\*=** (int \_\_rhs)
- decimal32 & **operator\*=** (unsigned int \_\_rhs)
- decimal32 & **operator\*=** (long \_\_rhs)
- decimal32 & **operator\*=** (unsigned long \_\_rhs)
- decimal32 & **operator\*=** (unsigned long long \_\_rhs)
- decimal32 & **operator\*=** (long long \_\_rhs)
- decimal32 & **operator++** ()
- decimal32 **operator++** (int)
- decimal32 & **operator+=** (int \_\_rhs)
- decimal32 & **operator+=** (decimal32 \_\_rhs)
- decimal32 & **operator+=** (decimal64 \_\_rhs)
- decimal32 & **operator+=** (decimal128 \_\_rhs)
- decimal32 & **operator+=** (unsigned int \_\_rhs)
- decimal32 & **operator+=** (long \_\_rhs)
- decimal32 & **operator+=** (unsigned long \_\_rhs)
- decimal32 & **operator+=** (long long \_\_rhs)
- decimal32 & **operator+=** (unsigned long long \_\_rhs)
- decimal32 & **operator--** ()
- decimal32 **operator--** (int)
- decimal32 & **operator-=** (long long \_\_rhs)
- decimal32 & **operator-=** (long \_\_rhs)
- decimal32 & **operator-=** (int \_\_rhs)
- decimal32 & **operator-=** (decimal32 \_\_rhs)
- decimal32 & **operator-=** (unsigned long \_\_rhs)
- decimal32 & **operator-=** (decimal128 \_\_rhs)
- decimal32 & **operator-=** (unsigned long long \_\_rhs)
- decimal32 & **operator-=** (unsigned int \_\_rhs)
- decimal32 & **operator-=** (decimal64 \_\_rhs)
- decimal32 & **operator/=** (decimal64 \_\_rhs)
- decimal32 & **operator/=** (long \_\_rhs)
- decimal32 & **operator/=** (long long \_\_rhs)
- decimal32 & **operator/=** (decimal32 \_\_rhs)
- decimal32 & **operator/=** (unsigned int \_\_rhs)
- decimal32 & **operator/=** (unsigned long \_\_rhs)
- decimal32 & **operator/=** (unsigned long long \_\_rhs)
- decimal32 & **operator/=** (decimal128 \_\_rhs)
- decimal32 & **operator/=** (int \_\_rhs)

### 5.441.1 Detailed Description

3.2.2 Class [decimal32](#).

Definition at line 227 of file decimal.

### 5.441.2 Constructor & Destructor Documentation

#### 5.441.2.1 std::decimal::decimal32::decimal32 ( \_\_decfloat32 \_\_z ) [inline]

Conforming extension: Conversion from scalar decimal type.

Definition at line 251 of file decimal.

The documentation for this class was generated from the following file:

- [decimal](#)

## 5.442 std::decimal::decimal64 Class Reference

3.2.3 Class [decimal64](#).

### Public Types

- typedef float \_\_decfloat64 \_\_attribute\_\_((mode(DD)))

### Public Member Functions

- **decimal64** ([decimal32](#) d32)
- **decimal64** (float \_\_r)
- **decimal64** (unsigned int \_\_z)
- **decimal64** (long \_\_z)
- **decimal64** (double \_\_r)
- **decimal64** (unsigned long \_\_z)
- **decimal64** (long long \_\_z)
- **decimal64** ([decimal128](#) d128)
- **decimal64** (long double \_\_r)
- **decimal64** (unsigned long long \_\_z)
- [decimal64](#) (\_\_decfloat64 \_\_z)
- **decimal64** (int \_\_z)
- \_\_decfloat64 \_\_getval (void)

- void **\_\_setval** (\_\_decfloat64 \_\_x)
- decimal64 & **operator\*=** (decimal32 \_\_rhs)
- decimal64 & **operator\*=** (decimal64 \_\_rhs)
- decimal64 & **operator\*=** (decimal128 \_\_rhs)
- decimal64 & **operator\*=** (int \_\_rhs)
- decimal64 & **operator\*=** (unsigned int \_\_rhs)
- decimal64 & **operator\*=** (long \_\_rhs)
- decimal64 & **operator\*=** (unsigned long \_\_rhs)
- decimal64 & **operator\*=** (unsigned long long \_\_rhs)
- decimal64 & **operator\*=** (long long \_\_rhs)
- decimal64 & **operator++** ()
- decimal64 **operator++** (int)
- decimal64 & **operator+=** (int \_\_rhs)
- decimal64 & **operator+=** (decimal32 \_\_rhs)
- decimal64 & **operator+=** (decimal64 \_\_rhs)
- decimal64 & **operator+=** (decimal128 \_\_rhs)
- decimal64 & **operator+=** (unsigned int \_\_rhs)
- decimal64 & **operator+=** (long \_\_rhs)
- decimal64 & **operator+=** (unsigned long \_\_rhs)
- decimal64 & **operator+=** (long long \_\_rhs)
- decimal64 & **operator+=** (unsigned long long \_\_rhs)
- decimal64 & **operator--** ()
- decimal64 **operator--** (int)
- decimal64 & **operator-=** (long long \_\_rhs)
- decimal64 & **operator-=** (long \_\_rhs)
- decimal64 & **operator-=** (int \_\_rhs)
- decimal64 & **operator-=** (decimal32 \_\_rhs)
- decimal64 & **operator-=** (unsigned long \_\_rhs)
- decimal64 & **operator-=** (decimal128 \_\_rhs)
- decimal64 & **operator-=** (unsigned long long \_\_rhs)
- decimal64 & **operator-=** (unsigned int \_\_rhs)
- decimal64 & **operator-=** (decimal64 \_\_rhs)
- decimal64 & **operator/=** (decimal64 \_\_rhs)
- decimal64 & **operator/=** (long \_\_rhs)
- decimal64 & **operator/=** (long long \_\_rhs)
- decimal64 & **operator/=** (decimal32 \_\_rhs)
- decimal64 & **operator/=** (unsigned int \_\_rhs)
- decimal64 & **operator/=** (unsigned long \_\_rhs)
- decimal64 & **operator/=** (unsigned long long \_\_rhs)
- decimal64 & **operator/=** (decimal128 \_\_rhs)
- decimal64 & **operator/=** (int \_\_rhs)

### 5.442.1 Detailed Description

3.2.3 Class [decimal64](#).

Definition at line 310 of file `decimal`.

### 5.442.2 Constructor & Destructor Documentation

#### 5.442.2.1 `std::decimal::decimal64::decimal64 ( __decfloat64 __z )` `[inline]`

Conforming extension: Conversion from scalar decimal type.

Definition at line 334 of file `decimal`.

The documentation for this class was generated from the following file:

- [decimal](#)

## 5.443 `std::default_delete<_Tp>` Struct Template Reference

Primary template, [default\\_delete](#).

### Public Member Functions

- `template<typename _Up , typename = typename std::enable_if<std::is_convertible<_Up*, _Tp*>::value>::type>`  
`default_delete (const default\_delete<_Up> &)`
- `void operator() (_Tp *__ptr) const`

### 5.443.1 Detailed Description

`template<typename _Tp> struct std::default_delete<_Tp>`

Primary template, [default\\_delete](#).

Definition at line 50 of file `unique_ptr.h`.

The documentation for this struct was generated from the following file:

- [unique\\_ptr.h](#)

### 5.444 `std::default_delete< _Tp[]>` Struct Template Reference

Specialization, [default\\_delete](#).

#### Public Member Functions

- `void operator() (_Tp *__ptr) const`

#### 5.444.1 Detailed Description

`template<typename _Tp> struct std::default_delete< _Tp[]>`

Specialization, [default\\_delete](#).

Definition at line 71 of file `unique_ptr.h`.

The documentation for this struct was generated from the following file:

- [unique\\_ptr.h](#)

### 5.445 `std::defer_lock_t` Struct Reference

Do not acquire ownership of the mutex.

#### 5.445.1 Detailed Description

Do not acquire ownership of the mutex.

Definition at line 422 of file `mutex`.

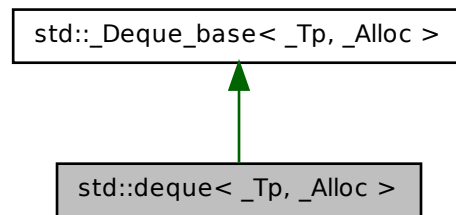
The documentation for this struct was generated from the following file:

- [mutex](#)

### 5.446 `std::deque< _Tp, _Alloc >` Class Template Reference

A standard container using fixed-size memory allocation and constant-time manipulation of elements at either end.

Inheritance diagram for `std::deque< _Tp, _Alloc >`:



### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_Base::const_iterator` **const\_iterator**
- typedef `_Tp_alloc_type::const_pointer` **const\_pointer**
- typedef `_Tp_alloc_type::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `ptrdiff_t` **difference\_type**
- typedef `_Base::iterator` **iterator**
- typedef `_Tp_alloc_type::pointer` **pointer**
- typedef `_Tp_alloc_type::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `size_t` **size\_type**
- typedef `_Tp` **value\_type**

### Public Member Functions

- `deque()`
- `deque(const allocator_type &__a)`
- `deque(size_type __n, const value_type &__value, const allocator_type &__a=allocator_type())`
- `template<typename _InputIterator >`  
`deque(_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())`
- `deque(const deque &__x)`
- `deque(size_type __n)`

- `deque` (`deque` &&\_\_x)
- `deque` (`initializer_list`< value\_type > \_\_l, const allocator\_type &\_\_a=allocator\_type())
- `~deque` ()
- void `assign` (size\_type \_\_n, const value\_type &\_\_val)
- template<typename \_InputIterator >  
void `assign` (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void `assign` (`initializer_list`< value\_type > \_\_l)
- reference `at` (size\_type \_\_n)
- const\_reference `at` (size\_type \_\_n) const
- reference `back` ()
- const\_reference `back` () const
- `iterator begin` ()
- `const_iterator begin` () const
- `const_iterator cbegin` () const
- `const_iterator cend` () const
- void `clear` ()
- `const_reverse_iterator crbegin` () const
- `const_reverse_iterator crend` () const
- template<typename... \_Args>  
`iterator emplace` (`iterator` \_\_position, \_Args &&...\_\_args)
- template<typename... \_Args>  
void `emplace_back` (\_Args &&...\_\_args)
- template<typename... \_Args>  
void `emplace_front` (\_Args &&...\_\_args)
- bool `empty` () const
- `iterator end` ()
- `const_iterator end` () const
- `iterator erase` (`iterator` \_\_position)
- `iterator erase` (`iterator` \_\_first, `iterator` \_\_last)
- const\_reference `front` () const
- reference `front` ()
- allocator\_type `get_allocator` () const
- `iterator insert` (`iterator` \_\_position, value\_type &&\_\_x)
- void `insert` (`iterator` \_\_p, `initializer_list`< value\_type > \_\_l)
- template<typename \_InputIterator >  
void `insert` (`iterator` \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- void `insert` (`iterator` \_\_position, size\_type \_\_n, const value\_type &\_\_x)
- `iterator insert` (`iterator` \_\_position, const value\_type &\_\_x)
- size\_type `max_size` () const
- `deque` & `operator=` (`initializer_list`< value\_type > \_\_l)
- `deque` & `operator=` (const `deque` &\_\_x)
- `deque` & `operator=` (`deque` &&\_\_x)



- reference `operator[]` (size\_type \_\_n)
- const\_reference `operator[]` (size\_type \_\_n) const
- void `pop_back` ()
- void `pop_front` ()
- void `push_back` (value\_type &&\_\_x)
- void `push_back` (const value\_type &\_\_x)
- void `push_front` (value\_type &&\_\_x)
- void `push_front` (const value\_type &\_\_x)
- `reverse_iterator` `rbegin` ()
- `const_reverse_iterator` `rbegin` () const
- `reverse_iterator` `rend` ()
- `const_reverse_iterator` `rend` () const
- void `resize` (size\_type \_\_new\_size)
- void `resize` (size\_type \_\_new\_size, const value\_type &\_\_x)
- void `shrink_to_fit` ()
- size\_type `size` () const
- void `swap` (deque &\_\_x)

### Protected Types

- enum { `_S_initial_map_size` }
- typedef `_Alloc::template rebind< _Tp * >::other` `_Map_alloc_type`
- typedef pointer \* `_Map_pointer`

### Protected Member Functions

- `_Tp **` `_M_allocate_map` (size\_t \_\_n)
- `_Tp *` `_M_allocate_node` ()
- template<typename `_ForwardIterator` >  
void `_M_assign_aux` (`_ForwardIterator` \_\_first, `_ForwardIterator` \_\_last, `std::forward_iterator_tag`)
- template<typename `_InputIterator` >  
void `_M_assign_aux` (`_InputIterator` \_\_first, `_InputIterator` \_\_last, `std::input_iterator_tag`)
- template<typename `_Integer` >  
void `_M_assign_dispatch` (`_Integer` \_\_n, `_Integer` \_\_val, \_\_true\_type)
- template<typename `_InputIterator` >  
void `_M_assign_dispatch` (`_InputIterator` \_\_first, `_InputIterator` \_\_last, \_\_false\_type)
- void `_M_create_nodes` (`_Tp **` \_\_nstart, `_Tp **` \_\_nfinish)
- void `_M_deallocate_map` (`_Tp **` \_\_p, size\_t \_\_n)
- void `_M_deallocate_node` (`_Tp *` \_\_p)
- void `_M_default_append` (size\_type \_\_n)

- `void _M_default_initialize ()`
- `template<typename _Alloc1 >`  
`void _M_destroy_data (iterator __first, iterator __last, const _Alloc1 &)`
- `void _M_destroy_data (iterator __first, iterator __last, const std::allocator<_Tp> &)`
- `void _M_destroy_data_aux (iterator __first, iterator __last)`
- `void _M_destroy_nodes (_Tp ** __nstart, _Tp ** __nfinish)`
- `void _M_erase_at_begin (iterator __pos)`
- `void _M_erase_at_end (iterator __pos)`
- `void _M_fill_assign (size_type __n, const value_type &__val)`
- `void _M_fill_initialize (const value_type &__value)`
- `void _M_fill_insert (iterator __pos, size_type __n, const value_type &__x)`
- `_Map_alloc_type _M_get_map_allocator () const`
- `_Tp_alloc_type & _M_get_Tp_allocator ()`
- `const _Tp_alloc_type & _M_get_Tp_allocator () const`
- `template<typename _Integer >`  
`void _M_initialize_dispatch (_Integer __n, _Integer __x, __true_type)`
- `template<typename _InputIterator >`  
`void _M_initialize_dispatch (_InputIterator __first, _InputIterator __last, __false_type)`
- `void _M_initialize_map (size_t)`
- `template<typename... _Args>`  
`iterator _M_insert_aux (iterator __pos, _Args &&... __args)`
- `template<typename _ForwardIterator >`  
`void _M_insert_aux (iterator __pos, _ForwardIterator __first, _ForwardIterator __last, size_type __n)`
- `void _M_insert_aux (iterator __pos, size_type __n, const value_type &__x)`
- `template<typename _Integer >`  
`void _M_insert_dispatch (iterator __pos, _Integer __n, _Integer __x, __true_type)`
- `template<typename _InputIterator >`  
`void _M_insert_dispatch (iterator __pos, _InputIterator __first, _InputIterator __last, __false_type)`
- `void _M_range_check (size_type __n) const`
- `template<typename _ForwardIterator >`  
`void _M_range_insert_aux (iterator __pos, _ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)`
- `template<typename _InputIterator >`  
`void _M_range_insert_aux (iterator __pos, _InputIterator __first, _InputIterator __last, std::input_iterator_tag)`
- `template<typename _InputIterator >`  
`void _M_range_initialize (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)`

- `template<typename _ForwardIterator >`  
`void _M_range_initialize (_ForwardIterator __first, _ForwardIterator __last,`  
`std::forward_iterator_tag)`
- `template<typename... _Args>`  
`void _M_push_back_aux (_Args &&...__args)`
- `template<typename... _Args>`  
`void _M_push_front_aux (_Args &&...__args)`
- `void _M_pop_back_aux ()`
- `void _M_pop_front_aux ()`
- `iterator _M_reserve_elements_at_front (size_type __n)`
- `iterator _M_reserve_elements_at_back (size_type __n)`
- `void _M_new_elements_at_front (size_type __new_elements)`
- `void _M_new_elements_at_back (size_type __new_elements)`
- `void _M_reserve_map_at_back (size_type __nodes_to_add=1)`
- `void _M_reserve_map_at_front (size_type __nodes_to_add=1)`
- `void _M_reallocate_map (size_type __nodes_to_add, bool __add_at_front)`

#### Static Protected Member Functions

- `static size_t _S_buffer_size ()`

#### Protected Attributes

- `_Deque_impl _M_impl`

#### 5.446.1 Detailed Description

`template<typename _Tp, typename _Alloc = std::allocator<_Tp>> class std::deque<_Tp, _Alloc>`

A standard container using fixed-size memory allocation and constant-time manipulation of elements at either end. Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#), including the [optional sequence requirements](#).

In previous HP/SGI versions of deque, there was an extra template parameter so users could control the node size. This extension turned out to violate the C++ standard (it can be detected using template template parameters), and it was removed.

Here's how a `deque<Tp>` manages memory. Each deque has 4 members:

- `Tp** _M_map`

- `size_t _M_map_size`
- `iterator _M_start, _M_finish`

`map_size` is at least 8. `map` is an array of `map_size` pointers-to-. (The name `map` has nothing to do with the `std::map` class, and **nodes** should not be confused with `std::list`'s usage of *node*.)

A *node* has no specific type name as such, but it is referred to as *node* in this file. It is a simple array-of-`Tp`. If `Tp` is very large, there will be one `Tp` element per node (i.e., an array of one). For non-huge `Tp`'s, node size is inversely related to `Tp` size: the larger the `Tp`, the fewer `Tp`'s will fit in a node. The goal here is to keep the total size of a node relatively small and constant over different `Tp`'s, to improve allocator efficiency.

Not every pointer in the `map` array will point to a node. If the initial number of elements in the deque is small, the /middle/ `map` pointers will be valid, and the ones at the edges will be unused. This same situation will arise as the `map` grows: available `map` pointers, if any, will be on the ends. As new nodes are created, only a subset of the `map`'s pointers need to be copied *outward*.

Class invariants:

- For any nonsingular iterator `i`:
  - `i.node` points to a member of the `map` array. (Yes, you read that correctly: `i.node` does not actually point to a node.) The member of the `map` array is what actually points to the node.
  - `i.first == *(i.node)` (This points to the node (first `Tp` element).)
  - `i.last == i.first + node_size`
  - `i.cur` is a pointer in the range `[i.first, i.last)`. NOTE: the implication of this is that `i.cur` is always a dereferenceable pointer, even if `i` is a past-the-end iterator.
- `Start` and `Finish` are always nonsingular iterators. NOTE: this means that an empty deque must have one node, a deque with  $<N$  elements (where  $N$  is the node buffer size) must have one node, a deque with  $N$  through  $(2N-1)$  elements must have two nodes, etc.
- For every node other than `start.node` and `finish.node`, every element in the node is an initialized object. If `start.node == finish.node`, then `[start.cur, finish.cur)` are initialized objects, and the elements outside that range are uninitialized storage. Otherwise, `[start.cur, start.last)` and `[finish.first, finish.cur)` are initialized objects, and `[start.first, start.cur)` and `[finish.cur, finish.last)` are uninitialized storage.
- `[map, map + map_size)` is a valid, non-empty range.
- `[start.node, finish.node]` is a valid range contained within `[map, map + map_size)`.

- A pointer in the range `[map, map + map_size)` points to an allocated node if and only if the pointer is in the range `[start.node, finish.node]`.

Here's the magic: nothing in `deque` is **aware** of the discontinuous storage!

The memory setup and layout occurs in the parent, `_Base`, and the iterator class is entirely responsible for *leaping* from one node to the next. All the implementation routines for `deque` itself work only through the start and finish iterators. This keeps the routines simple and sane, and we can use other standard algorithms as well.

Definition at line 719 of file `stl_deque.h`.

### 5.446.2 Constructor & Destructor Documentation

#### 5.446.2.1 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::deque<_Tp, _Alloc>::deque ( ) [inline]`

Default constructor creates no elements.

Definition at line 771 of file `stl_deque.h`.

#### 5.446.2.2 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::deque<_Tp, _Alloc>::deque ( const allocator_type & __a ) [inline, explicit]`

Creates a deque with no elements.

##### Parameters

*a* An allocator object.

Definition at line 779 of file `stl_deque.h`.

#### 5.446.2.3 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::deque<_Tp, _Alloc>::deque ( size_type __n ) [inline, explicit]`

Creates a deque with default constructed elements.

##### Parameters

*n* The number of elements to initially create.

This constructor fills the deque with  $n$  default constructed elements.

Definition at line 791 of file `std_deque.h`.

```
5.446.2.4 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
 std::deque< _Tp, _Alloc >::deque (size_type __n, const value_type
 & __value, const allocator_type & __a = allocator_type())
 [inline]
```

Creates a deque with copies of an exemplar element.

#### Parameters

*n* The number of elements to initially create.

*value* An element to copy.

*a* An allocator.

This constructor fills the deque with  $n$  copies of *value*.

Definition at line 803 of file `std_deque.h`.

References `std::deque< _Tp, _Alloc >::_M_fill_initialize()`.

```
5.446.2.5 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
 std::deque< _Tp, _Alloc >::deque (const deque< _Tp, _Alloc > &
 __x) [inline]
```

Deque copy constructor.

#### Parameters

*x* A deque of identical element and allocator types.

The newly-created deque uses a copy of the allocation object used by *x*.

Definition at line 830 of file `std_deque.h`.

References `std::deque< _Tp, _Alloc >::begin()`, and `std::deque< _Tp, _Alloc >::end()`.

```
5.446.2.6 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
 std::deque< _Tp, _Alloc >::deque (deque< _Tp, _Alloc > && __x
) [inline]
```

Deque move constructor.

### Parameters

*x* A deque of identical element and allocator types.

The newly-created deque contains the exact contents of *x*. The contents of *x* are a valid, but unspecified deque.

Definition at line 844 of file `stl_deque.h`.

```
5.446.2.7 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
 std::deque<_Tp, _Alloc>::deque (initializer_list< value_type
 > __l, const allocator_type & __a = allocator_type())
 [inline]
```

Builds a deque from an initializer list.

### Parameters

*l* An [initializer\\_list](#).

*a* An allocator object.

Create a deque consisting of copies of the elements in the [initializer\\_list](#) *l*.

This will call the element type's copy constructor *N* times (where *N* is *l.size()*) and do no memory reallocation.

Definition at line 858 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::_M_range_initialize()`.

```
5.446.2.8 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
 template<typename _InputIterator> std::deque<_Tp, _Alloc
 >::deque (_InputIterator __first, _InputIterator __last, const
 allocator_type & __a = allocator_type()) [inline]
```

Builds a deque from a range.

### Parameters

*first* An input iterator.

*last* An input iterator.

*a* An allocator object.

Create a deque consisting of copies of the elements from [first, last).

If the iterators are forward, bidirectional, or random-access, then this will call the elements' copy constructor *N* times (where *N* is `distance(first,last)`) and do no memory reallocation. But if only input iterators are used, then this will do at most *2N* calls to the copy constructor, and  $\log N$  memory reallocations.

Definition at line 883 of file `stl_deque.h`.

**5.446.2.9** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
std::deque<_Tp, _Alloc>::~~deque ( ) [inline]`

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 897 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::begin()`, and `std::deque<_Tp, _Alloc>::end()`.

### 5.446.3 Member Function Documentation

**5.446.3.1** `template<typename _Tp, typename _Alloc> void  
deque::_M_fill_initialize ( const value_type & __value )  
[protected]`

Fills the deque with copies of *value*.

#### Parameters

*value* Initial value.

#### Returns

Nothing.

#### Precondition

`_M_start` and `_M_finish` have already been initialized, but none of the deque's elements have yet been constructed.

This function is called only when the user provides an explicit size (with or without an explicit exemplar value).



Definition at line 333 of file `deque.tcc`.

References `std::_Destroy()`.

Referenced by `std::deque< _Tp, _Alloc >::deque()`.

**5.446.3.2** `template<typename _Tp , typename _Alloc > void  
std::_Deque_base< _Tp, _Alloc >::_M_initialize_map ( size_t  
__num_elements ) [protected, inherited]`

Layout storage.

#### Parameters

*num\_elements* The count of T's for which to allocate space at first.

#### Returns

Nothing.

The initial underlying memory layout is a bit complicated...

Definition at line 574 of file `stl_deque.h`.

References `std::max()`.

Referenced by `std::deque< _Tp, _Alloc >::_M_range_initialize()`.

**5.446.3.3** `template<typename _Tp , typename _Alloc > void  
deque::_M_new_elements_at_back ( size_type __new_elements )  
[protected]`

Memory-handling helpers for the previous internal insert functions.

Definition at line 827 of file `deque.tcc`.

References `std::deque< _Tp, _Alloc >::_M_reserve_map_at_back()`, `std::deque< _Tp, _Alloc >::max_size()`, and `std::deque< _Tp, _Alloc >::size()`.

Referenced by `std::deque< _Tp, _Alloc >::_M_reserve_elements_at_back()`.

**5.446.3.4** `template<typename _Tp , typename _Alloc > void  
deque::_M_new_elements_at_front ( size_type __new_elements )  
[protected]`

Memory-handling helpers for the previous internal insert functions.

Definition at line 802 of file deque.tcc.

References std::deque< \_Tp, \_Alloc >::\_M\_reserve\_map\_at\_front(), std::deque< \_Tp, \_Alloc >::max\_size(), and std::deque< \_Tp, \_Alloc >::size().

Referenced by std::deque< \_Tp, \_Alloc >::\_M\_reserve\_elements\_at\_front().

#### 5.446.3.5 template<typename \_Tp , typename \_Alloc > void deque::\_M\_pop\_back\_aux ( ) [protected]

Helper functions for push\_\* and pop\_\*.

Definition at line 483 of file deque.tcc.

Referenced by std::deque< \_Tp, \_Alloc >::pop\_back().

#### 5.446.3.6 template<typename \_Tp , typename \_Alloc > void deque::\_M\_pop\_front\_aux ( ) [protected]

Helper functions for push\_\* and pop\_\*.

Definition at line 498 of file deque.tcc.

Referenced by std::deque< \_Tp, \_Alloc >::pop\_front().

#### 5.446.3.7 template<typename \_Tp , typename \_Alloc > template<typename... \_Args> void deque::\_M\_push\_back\_aux ( \_Args &&... \_\_args ) [protected]

Helper functions for push\_\* and pop\_\*.

Definition at line 417 of file deque.tcc.

Referenced by std::deque< \_Tp, \_Alloc >::push\_back().

#### 5.446.3.8 template<typename \_Tp , typename \_Alloc > template<typename... \_Args> void deque::\_M\_push\_front\_aux ( \_Args &&... \_\_args ) [protected]

Helper functions for push\_\* and pop\_\*.

Definition at line 451 of file `deque.tcc`.

Referenced by `std::deque< _Tp, _Alloc >::push_front()`.

```
5.446.3.9 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
 void std::deque< _Tp, _Alloc >::_M_range_check (size_type __n)
 const [inline, protected]
```

Safety check used only from `at()`.

Definition at line 1237 of file `std_deque.h`.

References `std::deque< _Tp, _Alloc >::size()`.

Referenced by `std::deque< _Tp, _Alloc >::at()`.

```
5.446.3.10 template<typename _Tp , typename _Alloc > template<typename
 _InputIterator > void deque::_M_range_initialize (_InputIterator
 __first, _InputIterator __last, std::input_iterator_tag)
 [protected]
```

Fills the deque with whatever is in `[first,last)`.

#### Parameters

*first* An input iterator.

*last* An input iterator.

#### Returns

Nothing.

If the iterators are actually forward iterators (or better), then the memory layout can be done all at once. Else we move forward using `push_back` on each value from the iterator.

Definition at line 359 of file `deque.tcc`.

References `std::_Deque_base< _Tp, _Alloc >::_M_initialize_map()`, `std::deque< _Tp, _Alloc >::clear()`, and `std::deque< _Tp, _Alloc >::push_back()`.

Referenced by `std::deque< _Tp, _Alloc >::deque()`.

**5.446.3.11** `template<typename _Tp, typename _Alloc > template<typename  
_ForwardIterator > void deque::_M_range_initialize (  
_ForwardIterator __first, _ForwardIterator __last,  
std::forward_iterator_tag ) [protected]`

Fills the deque with whatever is in [first,last).

#### Parameters

*first* An input iterator.

*last* An input iterator.

#### Returns

Nothing.

If the iterators are actually forward iterators (or better), then the memory layout can be done all at once. Else we move forward using push\_back on each value from the iterator.

Definition at line 379 of file deque.tcc.

References std::\_Destroy(), std::\_Deque\_base< \_Tp, \_Alloc >::\_M\_initialize\_map(), std::advance(), and std::distance().

**5.446.3.12** `template<typename _Tp, typename _Alloc > void  
deque::_M_reallocate_map ( size_type __nodes_to_add, bool  
__add_at_front ) [protected]`

Memory-handling helpers for the major map.

Makes sure the \_M\_map has space for new nodes. Does not actually add the nodes. Can invalidate \_M\_map pointers. (And consequently, deque iterators.)

Definition at line 852 of file deque.tcc.

References std::max().

Referenced by std::deque< \_Tp, \_Alloc >::\_M\_reserve\_map\_at\_back(), and std::deque< \_Tp, \_Alloc >::\_M\_reserve\_map\_at\_front().

**5.446.3.13** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
iterator std::deque< _Tp, _Alloc >::_M_reserve_elements_at_back  
( size_type __n ) [inline, protected]`

Memory-handling helpers for the previous internal insert functions.

Definition at line 1858 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::_M_new_elements_at_back()`.

**5.446.3.14** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
iterator std::deque< _Tp, _Alloc >::_M_reserve_elements_at_front  
( size_type __n ) [inline, protected]`

Memory-handling helpers for the previous internal insert functions.

Definition at line 1848 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::_M_new_elements_at_front()`.

**5.446.3.15** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::deque< _Tp, _Alloc >::_M_reserve_map_at_back (   
size_type __nodes_to_add = 1 ) [inline, protected]`

Memory-handling helpers for the major map.

Makes sure the `_M_map` has space for new nodes. Does not actually add the nodes. Can invalidate `_M_map` pointers. (And consequently, deque iterators.)

Definition at line 1884 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::_M_reallocate_map()`.

Referenced by `std::deque< _Tp, _Alloc >::_M_new_elements_at_back()`.

**5.446.3.16** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::deque< _Tp, _Alloc >::_M_reserve_map_at_front (   
size_type __nodes_to_add = 1 ) [inline, protected]`

Memory-handling helpers for the major map.

Makes sure the `_M_map` has space for new nodes. Does not actually add the nodes. Can invalidate `_M_map` pointers. (And consequently, deque iterators.)

Definition at line 1892 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::_M_reallocate_map()`.

Referenced by `std::deque< _Tp, _Alloc >::_M_new_elements_at_front()`.

**5.446.3.17** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::deque< _Tp, _Alloc >::assign ( size_type __n, const  
value_type & __val ) [inline]`

Assigns a given value to a deque.

#### Parameters

*n* Number of elements to be assigned.

*val* Value to be assigned.

This function fills a deque with *n* copies of the given value. Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 958 of file stl\_deque.h.

Referenced by std::deque< \_Tp, \_Alloc >::operator=().

**5.446.3.18** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
template<typename InputIterator > void std::deque< _Tp, _Alloc  
>::assign ( InputIterator __first, InputIterator __last )  
[inline]`

Assigns a range to a deque.

#### Parameters

*first* An input iterator.

*last* An input iterator.

This function fills a deque with copies of the elements in the range [first,last).

Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 975 of file stl\_deque.h.

**5.446.3.19** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::deque< _Tp, _Alloc >::assign ( initializer_list<  
value_type > __l ) [inline]`

Assigns an initializer list to a deque.

**Parameters**

*l* An [initializer\\_list](#).

This function fills a deque with copies of the elements in the [initializer\\_list](#) *l*.

Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 994 of file `std_deque.h`.

References `std::deque< _Tp, _Alloc >::assign()`.

Referenced by `std::deque< _Tp, _Alloc >::assign()`.

**5.446.3.20** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
reference std::deque< _Tp, _Alloc >::at ( size_type __n )  
[inline]`

Provides access to the data contained in the deque.

**Parameters**

*n* The index of the element for which data should be accessed.

**Returns**

Read/write reference to data.

**Exceptions**

[std::out\\_of\\_range](#) If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the deque. The function throws [out\\_of\\_range](#) if the check fails.

Definition at line 1256 of file `std_deque.h`.

References `std::deque< _Tp, _Alloc >::_M_range_check()`.

**5.446.3.21** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_reference std::deque< _Tp, _Alloc >::at ( size_type __n )  
const [inline]`

Provides access to the data contained in the deque.

**Parameters**

*n* The index of the element for which data should be accessed.

**Returns**

Read-only (constant) reference to data.

**Exceptions**

*std::out\_of\_range* If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the deque. The function throws *out\_of\_range* if the check fails.

Definition at line 1274 of file `std_deque.h`.

References `std::deque<_Tp, _Alloc>::_M_range_check()`.

**5.446.3.22** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_reference std::deque<_Tp, _Alloc>::back ( ) const  
[inline]`

Returns a read-only (constant) reference to the data at the last element of the deque.

Definition at line 1313 of file `std_deque.h`.

References `std::deque<_Tp, _Alloc>::end()`.

**5.446.3.23** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
reference std::deque<_Tp, _Alloc>::back ( ) [inline]`

Returns a read/write reference to the data at the last element of the deque.

Definition at line 1301 of file `std_deque.h`.

References `std::deque<_Tp, _Alloc>::end()`.

**5.446.3.24** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
iterator std::deque<_Tp, _Alloc>::begin ( ) [inline]`

Returns a read/write iterator that points to the first element in the deque. Iteration is done in ordinary element order.

Definition at line 1009 of file `std_deque.h`.

Referenced by `std::deque<_Tp, _Alloc>::clear()`, `std::deque<_Tp, _Alloc>::deque()`, `std::deque<_Tp, _Alloc>::erase()`, `std::deque<_Tp, _Alloc>::front()`, `std::deque<_Tp, _Alloc>::operator=()`, `std::operator==( )`, and `std::deque<_Tp, _Alloc>::~~deque()`.



**5.446.3.25** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_iterator std::deque<_Tp, _Alloc>::begin ( ) const  
[inline]`

Returns a read-only (constant) iterator that points to the first element in the deque. Iteration is done in ordinary element order.

Definition at line 1017 of file `stl_deque.h`.

**5.446.3.26** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_iterator std::deque<_Tp, _Alloc>::cbegin ( ) const  
[inline]`

Returns a read-only (constant) iterator that points to the first element in the deque. Iteration is done in ordinary element order.

Definition at line 1080 of file `stl_deque.h`.

**5.446.3.27** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_iterator std::deque<_Tp, _Alloc>::cend ( ) const  
[inline]`

Returns a read-only (constant) iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

Definition at line 1089 of file `stl_deque.h`.

**5.446.3.28** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::deque<_Tp, _Alloc>::clear ( ) [inline]`

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1579 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::begin()`.

Referenced by `std::deque<_Tp, _Alloc>::_M_range_initialize()`, `std::deque<_Tp, _Alloc>::erase()`, and `std::deque<_Tp, _Alloc>::operator=()`.

**5.446.3.29** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_reverse_iterator std::deque< _Tp, _Alloc >::crbegin ( )  
const [inline]`

Returns a read-only (constant) reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

Definition at line 1098 of file `stl_deque.h`.

**5.446.3.30** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_reverse_iterator std::deque< _Tp, _Alloc >::crend ( ) const  
[inline]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

Definition at line 1107 of file `stl_deque.h`.

**5.446.3.31** `template<typename _Tp, typename _Alloc > template<typename...  
_Args> deque< _Tp, _Alloc >::iterator deque::emplace ( iterator  
__position, _Args &&... __args )`

Inserts an object in deque before specified iterator.

#### Parameters

*position* An iterator into the deque.

*args* Arguments.

#### Returns

An iterator that points to the inserted data.

This function will insert an object of type `T` constructed with `T(std::forward<Args>(args)...) before the specified location.`

Definition at line 174 of file `deque.tcc`.

References `std::deque< _Tp, _Alloc >::push_back()`, and `std::deque< _Tp, _Alloc >::push_front()`.

Referenced by `std::deque< _Tp, _Alloc >::insert()`.

**5.446.3.32** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
bool std::deque<_Tp, _Alloc>::empty ( ) const [inline]`

Returns true if the deque is empty. (Thus `begin()` would equal `end()`.)

Definition at line 1200 of file `stl_deque.h`.

**5.446.3.33** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
iterator std::deque<_Tp, _Alloc>::end ( ) [inline]`

Returns a read/write iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

Definition at line 1026 of file `stl_deque.h`.

Referenced by `std::deque<_Tp, _Alloc>::back()`, `std::deque<_Tp, _Alloc>::deque()`, `std::deque<_Tp, _Alloc>::erase()`, `std::deque<_Tp, _Alloc>::operator=()`, `std::operator==()`, and `std::deque<_Tp, _Alloc>::~~deque()`.

**5.446.3.34** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_iterator std::deque<_Tp, _Alloc>::end ( ) const  
[inline]`

Returns a read-only (constant) iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

Definition at line 1035 of file `stl_deque.h`.

**5.446.3.35** `template<typename _Tp, typename _Alloc> deque<_Tp, _Alloc>  
>::iterator deque::erase ( iterator __position )`

Remove element at given position.

#### Parameters

*position* Iterator pointing to element to be erased.

#### Returns

An iterator pointing to the next element (or `end()`).

This function will erase the element at the given position and thus shorten the deque by one.

The user is cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 196 of file `deque.tcc`.

References `std::deque< _Tp, _Alloc >::begin()`, `std::deque< _Tp, _Alloc >::end()`, `std::deque< _Tp, _Alloc >::pop_back()`, `std::deque< _Tp, _Alloc >::pop_front()`, and `std::deque< _Tp, _Alloc >::size()`.

### 5.446.3.36 `template<typename _Tp, typename _Alloc > deque< _Tp, _Alloc >::iterator deque::erase ( iterator __first, iterator __last )`

Remove a range of elements.

#### Parameters

*first* Iterator pointing to the first element to be erased.

*last* Iterator pointing to one past the last element to be erased.

#### Returns

An iterator pointing to the element pointed to by *last* prior to erasing (or `end()`).

This function will erase the elements in the range `[first,last)` and shorten the deque accordingly.

The user is cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 219 of file `deque.tcc`.

References `std::deque< _Tp, _Alloc >::begin()`, `std::deque< _Tp, _Alloc >::clear()`, `std::deque< _Tp, _Alloc >::end()`, and `std::deque< _Tp, _Alloc >::size()`.

### 5.446.3.37 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::deque< _Tp, _Alloc >::front ( ) [inline]`

Returns a read/write reference to the data at the first element of the deque.

Definition at line 1285 of file `std_deque.h`.

References `std::deque< _Tp, _Alloc >::begin()`.

**5.446.3.38** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_reference std::deque< _Tp, _Alloc >::front ( ) const  
[inline]`

Returns a read-only (constant) reference to the data at the first element of the deque.

Definition at line 1293 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::begin()`.

**5.446.3.39** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
allocator_type std::deque< _Tp, _Alloc >::get_allocator ( ) const  
[inline]`

Get a copy of the memory allocation object.

Reimplemented from `std::_Deque_base< _Tp, _Alloc >`.

Definition at line 1000 of file `stl_deque.h`.

**5.446.3.40** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
template<typename _InputIterator > void std::deque< _Tp,  
_Alloc >::insert ( iterator __position, _InputIterator __first,  
_InputIterator __last ) [inline]`

Inserts a range into the deque.

#### Parameters

*position* An iterator into the deque.

*first* An input iterator.

*last* An input iterator.

This function will insert copies of the data in the range `[first,last)` into the deque before the location specified by *pos*. This is known as *range insert*.

Definition at line 1506 of file `stl_deque.h`.

**5.446.3.41** `template<typename _Tp , typename _Alloc > deque< _Tp, _Alloc  
>::iterator deque::insert ( iterator __position, const value_type &  
__x )`

Inserts given value into deque before specified iterator.

### Parameters

*position* An iterator into the deque.  
*x* Data to be inserted.

### Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location.

Definition at line 151 of file deque.tcc.

References std::deque< \_Tp, \_Alloc >::push\_back(), and std::deque< \_Tp, \_Alloc >::push\_front().

Referenced by std::deque< \_Tp, \_Alloc >::operator=(), and std::deque< \_Tp, \_Alloc >::resize().

**5.446.3.42** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
 void std::deque< _Tp, _Alloc >::insert ( iterator __p,  
 initializer_list< value_type > __l ) [inline]`

Inserts an initializer list into the deque.

### Parameters

*p* An iterator into the deque.  
*l* An [initializer\\_list](#).

This function will insert copies of the data in the [initializer\\_list](#) *l* into the deque before the location specified by *p*. This is known as *list insert*.

Definition at line 1477 of file stl\_deque.h.

References std::deque< \_Tp, \_Alloc >::insert().

Referenced by std::deque< \_Tp, \_Alloc >::insert().

**5.446.3.43** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
 iterator std::deque< _Tp, _Alloc >::insert ( iterator __position,  
 value_type && __x ) [inline]`

Inserts given rvalue into deque before specified iterator.

**Parameters**

*position* An iterator into the deque.  
*x* Data to be inserted.

**Returns**

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location.

Definition at line 1464 of file stl\_deque.h.

References std::deque< \_Tp, \_Alloc >::emplace().

**5.446.3.44** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
 void std::deque< _Tp, _Alloc >::insert ( iterator __position,  
 size_type __n, const value_type & __x ) [inline]`

Inserts a number of copies of given data into the deque.

**Parameters**

*position* An iterator into the deque.  
*n* Number of elements to be inserted.  
*x* Data to be inserted.

This function will insert a specified number of copies of the given data before the location specified by *position*.

Definition at line 1491 of file stl\_deque.h.

**5.446.3.45** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
 size_type std::deque< _Tp, _Alloc >::max_size ( ) const  
 [inline]`

Returns the [size\(\)](#) of the largest possible deque.

Definition at line 1119 of file stl\_deque.h.

Referenced by std::deque< \_Tp, \_Alloc >::\_M\_new\_elements\_at\_back(), and std::deque< \_Tp, \_Alloc >::\_M\_new\_elements\_at\_front().

**5.446.3.46** `template<typename _Tp, typename _Alloc > deque< _Tp, _Alloc  
 > & deque::operator= ( const deque< _Tp, _Alloc > & __x )`

Deque assignment operator.

#### Parameters

*x* A deque of identical element and allocator types.

All the elements of *x* are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 95 of file `deque.tcc`.

References `std::deque<_Tp, _Alloc>::begin()`, `std::deque<_Tp, _Alloc>::end()`, `std::deque<_Tp, _Alloc>::insert()`, and `std::deque<_Tp, _Alloc>::size()`.

**5.446.3.47** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
deque& std::deque<_Tp, _Alloc>::operator= ( initializer_list<  
value_type > __l ) [inline]`

Assigns an initializer list to a deque.

#### Parameters

*l* An [initializer\\_list](#).

This function fills a deque with copies of the elements in the [initializer\\_list](#) *l*.

Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 940 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::assign()`.

**5.446.3.48** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
deque& std::deque<_Tp, _Alloc>::operator= ( deque<_Tp,  
_Alloc> && __x ) [inline]`

Deque move assignment operator.

#### Parameters

*x* A deque of identical element and allocator types.

The contents of *x* are moved into this deque (without copying). *x* is a valid, but unspecified deque.



Definition at line 919 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::clear()`, and `std::deque<_Tp, _Alloc>::swap()`.

**5.446.3.49** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
reference std::deque<_Tp, _Alloc>::operator[] ( size_type __n )  
[inline]`

Subscript access to the data contained in the deque.

#### Parameters

*n* The index of the element for which data should be accessed.

#### Returns

Read/write reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and [out\\_of\\_range](#) lookups are not defined. (For checked lookups see [at\(\)](#).)

Definition at line 1216 of file `stl_deque.h`.

**5.446.3.50** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_reference std::deque<_Tp, _Alloc>::operator[] ( size_type  
__n ) const [inline]`

Subscript access to the data contained in the deque.

#### Parameters

*n* The index of the element for which data should be accessed.

#### Returns

Read-only (constant) reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and [out\\_of\\_range](#) lookups are not defined. (For checked lookups see [at\(\)](#).)

Definition at line 1231 of file `stl_deque.h`.

**5.446.3.51** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::deque<_Tp, _Alloc>::pop_back ( ) [inline]`

Removes last element.

This is a typical stack operation. It shrinks the deque by one.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before `pop_back()` is called.

Definition at line 1414 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::_M_pop_back_aux()`.

Referenced by `std::deque<_Tp, _Alloc>::erase()`.

**5.446.3.52** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::deque<_Tp, _Alloc>::pop_front ( ) [inline]`

Removes first element.

This is a typical stack operation. It shrinks the deque by one.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop_front()` is called.

Definition at line 1393 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::_M_pop_front_aux()`.

Referenced by `std::deque<_Tp, _Alloc>::erase()`.

**5.446.3.53** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::deque<_Tp, _Alloc>::push_back ( const value_type &  
__x ) [inline]`

Add data to the end of the deque.

#### Parameters

*x* Data to be added.

This is a typical stack operation. The function creates an element at the end of the deque and assigns the given data to it. Due to the nature of a deque this operation can be done in constant time.

Definition at line 1362 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::_M_push_back_aux()`.

Referenced by `std::deque< _Tp, _Alloc >::_M_range_initialize()`, `std::deque< _Tp, _Alloc >::emplace()`, and `std::deque< _Tp, _Alloc >::insert()`.

**5.446.3.54** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::deque< _Tp, _Alloc >::push_front ( const value_type &  
__x ) [inline]`

Add data to the front of the deque.

#### Parameters

*x* Data to be added.

This is a typical stack operation. The function creates an element at the front of the deque and assigns the given data to it. Due to the nature of a deque this operation can be done in constant time.

Definition at line 1331 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::_M_push_front_aux()`.

Referenced by `std::deque< _Tp, _Alloc >::emplace()`, and `std::deque< _Tp, _Alloc >::insert()`.

**5.446.3.55** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_reverse_iterator std::deque< _Tp, _Alloc >::rbegin ( ) const  
[inline]`

Returns a read-only (constant) reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

Definition at line 1053 of file `stl_deque.h`.

**5.446.3.56** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
reverse_iterator std::deque< _Tp, _Alloc >::rbegin ( )  
[inline]`

Returns a read/write reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

Definition at line 1044 of file `stl_deque.h`.

**5.446.3.57** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
reverse_iterator std::deque< _Tp, _Alloc >::rend ( ) [inline]`

Returns a read/write reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

Definition at line 1062 of file stl\_deque.h.

**5.446.3.58** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_reverse_iterator std::deque< _Tp, _Alloc >::rend ( ) const  
[inline]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

Definition at line 1071 of file stl\_deque.h.

**5.446.3.59** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::deque< _Tp, _Alloc >::resize ( size_type __new_size )  
[inline]`

Resizes the deque to the specified number of elements.

#### Parameters

*new\_size* Number of elements the deque should contain.

This function will resize the deque to the specified number of elements. If the number is smaller than the deque's current size the deque is truncated, otherwise default constructed elements are appended.

Definition at line 1133 of file stl\_deque.h.

References `std::deque< _Tp, _Alloc >::size()`.

**5.446.3.60** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::deque< _Tp, _Alloc >::resize ( size_type __new_size,  
const value_type & __x ) [inline]`

Resizes the deque to the specified number of elements.

#### Parameters

*new\_size* Number of elements the deque should contain.

*x* Data with which new elements should be populated.

This function will resize the deque to the specified number of elements. If the number is smaller than the deque's current size the deque is truncated, otherwise the deque is extended and new elements are populated with given data.

Definition at line 1155 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::insert()`, and `std::deque< _Tp, _Alloc >::size()`.

**5.446.3.61** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::deque< _Tp, _Alloc >::shrink_to_fit ( ) [inline]`

A non-binding request to reduce memory use.

Definition at line 1191 of file `stl_deque.h`.

**5.446.3.62** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
size_type std::deque< _Tp, _Alloc >::size ( ) const [inline]`

Returns the number of elements in the deque.

Definition at line 1114 of file `stl_deque.h`.

Referenced by `std::deque< _Tp, _Alloc >::_M_new_elements_at_back()`, `std::deque< _Tp, _Alloc >::_M_new_elements_at_front()`, `std::deque< _Tp, _Alloc >::_M_range_check()`, `std::deque< _Tp, _Alloc >::erase()`, `std::deque< _Tp, _Alloc >::operator=()`, `std::operator==()`, and `std::deque< _Tp, _Alloc >::resize()`.

**5.446.3.63** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::deque< _Tp, _Alloc >::swap ( deque< _Tp, _Alloc > &  
__x ) [inline]`

Swaps data with another deque.

### Parameters

*x* A deque of the same element and allocator types.

This exchanges the elements between two deques in constant time. (Four pointers, so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(d1,d2)` will feed to this function.

Definition at line 1559 of file `stl_deque.h`.

Referenced by `std::deque< _Tp, _Alloc >::operator=()`, and `std::swap()`.

The documentation for this class was generated from the following files:

- [stl\\_deque.h](#)
- [deque.tcc](#)

## **5.447 std::discard\_block\_engine< \_RandomNumberEngine, \_\_p, \_\_r > Class Template Reference**

### **Public Types**

- `typedef _RandomNumberEngine::result_type` [result\\_type](#)

### **Public Member Functions**

- [discard\\_block\\_engine](#) ()
- [discard\\_block\\_engine](#) (const \_RandomNumberEngine &\_\_rne)
- [discard\\_block\\_engine](#) (result\_type \_\_s)
- `template<typename _Sseq, typename = typename std::enable_if<!std::is_same<_Sseq, discard_block_engine>::value && !std::is_same<_Sseq, _RandomNumberEngine>::value>::type>`  
[discard\\_block\\_engine](#) (\_Sseq &\_\_q)
- [discard\\_block\\_engine](#) (\_RandomNumberEngine &&\_\_rne)
- `const _RandomNumberEngine &` [base](#) () `const`
- `void` [discard](#) (unsigned long long \_\_z)
- `result_type` [operator](#)() ()
- `void` [seed](#) ()
- `template<typename _Sseq >`  
`void` [seed](#) (\_Sseq &\_\_q)
- `void` [seed](#) (result\_type \_\_s)

### **Static Public Member Functions**

- `static constexpr` [result\\_type](#) [max](#) ()
- `static constexpr` [result\\_type](#) [min](#) ()

### **Static Public Attributes**

- `static constexpr size_t` [block\\_size](#)
- `static constexpr size_t` [used\\_block](#)

## Friends

- `template<typename _RandomNumberEngine1, size_t __p1, size_t __r1, typename _CharT, typename _Traits>`  
`std::basic_ostream<_CharT, _Traits> & operator<< (std::basic_ostream<_CharT, _Traits> &, const std::discard_block_engine<_RandomNumberEngine1, __p1, __r1> &)`
- `bool operator== (const discard_block_engine &__lhs, const discard_block_engine &__rhs)`
- `template<typename _RandomNumberEngine1, size_t __p1, size_t __r1, typename _CharT, typename _Traits>`  
`std::basic_istream<_CharT, _Traits> & operator>> (std::basic_istream<_CharT, _Traits> &, std::discard_block_engine<_RandomNumberEngine1, __p1, __r1> &)`

### 5.447.1 Detailed Description

`template<typename _RandomNumberEngine, size_t __p, size_t __r> class std::discard_block_engine<_RandomNumberEngine, __p, __r>`

Produces random numbers from some base engine by discarding blocks of data.

`0 <= __r <= __p`

Definition at line 779 of file random.h.

### 5.447.2 Member Typedef Documentation

**5.447.2.1** `template<typename _RandomNumberEngine, size_t __p, size_t __r> typedef _RandomNumberEngine::result_type std::discard_block_engine<_RandomNumberEngine, __p, __r>::result_type`

The type of the generated random value.

Definition at line 786 of file random.h.

### 5.447.3 Constructor & Destructor Documentation

**5.447.3.1** `template<typename _RandomNumberEngine, size_t __p, size_t __r> std::discard_block_engine<_RandomNumberEngine, __p, __r>::discard_block_engine( ) [inline]`

Constructs a default discard\_block\_engine engine.

The underlying engine is default constructed as well.

Definition at line 797 of file `random.h`.

**5.447.3.2** `template<typename _RandomNumberEngine, size_t __p, size_t  
__r> std::discard_block_engine<_RandomNumberEngine, __p, __r  
>::discard_block_engine ( const _RandomNumberEngine & __rne )  
[inline, explicit]`

Copy constructs a `discard_block_engine` engine.

Copies an existing base class random number generator.

#### Parameters

*rng* An existing (base class) engine object.

Definition at line 807 of file `random.h`.

**5.447.3.3** `template<typename _RandomNumberEngine, size_t __p, size_t  
__r> std::discard_block_engine<_RandomNumberEngine, __p, __r  
>::discard_block_engine ( _RandomNumberEngine && __rne )  
[inline, explicit]`

Move constructs a `discard_block_engine` engine.

Copies an existing base class random number generator.

#### Parameters

*rng* An existing (base class) engine object.

Definition at line 817 of file `random.h`.

**5.447.3.4** `template<typename _RandomNumberEngine, size_t __p, size_t  
__r> std::discard_block_engine<_RandomNumberEngine, __p,  
__r>::discard_block_engine ( result_type __s ) [inline,  
explicit]`

Seed constructs a `discard_block_engine` engine.

Constructs the underlying generator engine seeded with `__s`.



#### Parameters

`__s` A seed value for the base class engine.

Definition at line 827 of file random.h.

**5.447.3.5** `template<typename _RandomNumberEngine, size_t __p,  
size_t __r> template<typename _Sseq, typename = typename  
std::enable_if<!std::is_same<_Sseq, discard_block_engine>::value  
&& !std::is_same<_Sseq, _RandomNumberEngine>::value>  
::type> std::discard_block_engine< _RandomNumberEngine,  
__p, __r >::discard_block_engine ( _Sseq & __q ) [inline,  
explicit]`

Generator construct a `discard_block_engine` engine.

#### Parameters

`__q` A seed sequence.

Definition at line 840 of file random.h.

#### 5.447.4 Member Function Documentation

**5.447.4.1** `template<typename _RandomNumberEngine, size_t __p, size_t  
__r> const _RandomNumberEngine& std::discard_block_engine<  
_RandomNumberEngine, __p, __r >::base ( ) const [inline]`

Gets a const reference to the underlying generator engine object.

Definition at line 884 of file random.h.

**5.447.4.2** `template<typename _RandomNumberEngine, size_t __p, size_t __r>  
void std::discard_block_engine< _RandomNumberEngine, __p, __r  
>::discard ( unsigned long long __z ) [inline]`

Discard a sequence of random numbers.

Definition at line 905 of file random.h.

5.447.4.3 `template<typename _RandomNumberEngine, size_t __p, size_t  
__r> static constexpr result_type std::discard_block_engine<  
_RandomNumberEngine, __p, __r>::max ( ) [inline,  
static]`

Gets the maximum value in the generated random number range.

Definition at line 898 of file random.h.

5.447.4.4 `template<typename _RandomNumberEngine, size_t __p, size_t  
__r> static constexpr result_type std::discard_block_engine<  
_RandomNumberEngine, __p, __r>::min ( ) [inline,  
static]`

Gets the minimum value in the generated random number range.

Definition at line 891 of file random.h.

5.447.4.5 `template<typename _RandomNumberEngine, size_t __p, size_t  
__r> discard_block_engine<_RandomNumberEngine, __p, __r  
>::result_type std::discard_block_engine<_RandomNumberEngine,  
__p, __r>::operator() ( )`

Gets the next value in the generated random number sequence.

Definition at line 670 of file random.tcc.

5.447.4.6 `template<typename _RandomNumberEngine, size_t __p, size_t  
__r> template<typename _Sseq > void std::discard_block_engine<  
_RandomNumberEngine, __p, __r>::seed ( _Sseq & __q )  
[inline]`

Reseeds the `discard_block_engine` object with the given seed sequence.

#### Parameters

`__q` A seed generator function.

Definition at line 873 of file random.h.

**5.447.4.7** `template<typename _RandomNumberEngine, size_t __p, size_t __r>  
void std::discard_block_engine<_RandomNumberEngine, __p, __r  
>::seed ( result_type __s ) [inline]`

Reseeds the discard\_block\_engine object with the default seed for the underlying base class generator engine.

Definition at line 860 of file random.h.

**5.447.4.8** `template<typename _RandomNumberEngine, size_t __p, size_t __r>  
void std::discard_block_engine<_RandomNumberEngine, __p, __r  
>::seed ( ) [inline]`

Reseeds the discard\_block\_engine object with the default seed for the underlying base class generator engine.

Definition at line 849 of file random.h.

## 5.447.5 Friends And Related Function Documentation

**5.447.5.1** `template<typename _RandomNumberEngine, size_t __p,  
size_t __r> template<typename _RandomNumberEngine1 ,  
size_t __p1, size_t __r1, typename _CharT , typename _Traits  
> std::basic_ostream<_CharT, _Traits>& operator<<  
( std::basic_ostream<_CharT, _Traits> & , const  
std::discard_block_engine<_RandomNumberEngine1, __p1, __r1 >  
& ) [friend]`

Inserts the current state of a discard\_block\_engine random number generator engine \_\_x into the output stream \_\_os.

### Parameters

`__os` An output stream.

`__x` A discard\_block\_engine random number generator engine.

### Returns

The output stream with the state of \_\_x inserted or in an error state.

**5.447.5.2** `template<typename _RandomNumberEngine, size_t __p,  
size_t __r> bool operator==( const discard_block_engine<  
_RandomNumberEngine, __p, __r > & __lhs, const  
discard_block_engine< _RandomNumberEngine, __p, __r > &  
__rhs ) [friend]`

Compares two discard\_block\_engine random number generator objects of the same type for equality.

**Parameters**

`__lhs` A discard\_block\_engine random number generator object.  
`__rhs` Another discard\_block\_engine random number generator object.

**Returns**

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 929 of file random.h.

**5.447.5.3** `template<typename _RandomNumberEngine, size_t __p,  
size_t __r> template<typename _RandomNumberEngine1  
, size_t __p1, size_t __r1, typename _CharT , typename  
_Traits > std::basic_istream<_CharT, _Traits>&  
operator>> ( std::basic_istream< _CharT, _Traits > & ,  
std::discard_block_engine< _RandomNumberEngine1, __p1, __r1 >  
& ) [friend]`

Extracts the current state of a % subtract\_with\_carry\_engine random number generator engine `__x` from the input stream `__is`.

**Parameters**

`__is` An input stream.  
`__x` A discard\_block\_engine random number generator engine.

**Returns**

The input stream with the state of `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.448 `std::discrete_distribution<_IntType>` Class Template Reference

A [`discrete\_distribution`](#) random number distribution.

### Classes

- struct [`param\_type`](#)

### Public Types

- typedef `_IntType` [`result\_type`](#)

### Public Member Functions

- template<typename `_InputIterator`>  
**`discrete_distribution`** (`_InputIterator` \_\_wbegin, `_InputIterator` \_\_wend)
- template<typename `_Func`>  
**`discrete_distribution`** (`size_t` \_\_nw, `double` \_\_xmin, `double` \_\_xmax, `_Func` \_\_fw)
- **`discrete_distribution`** (const [`param\_type`](#) &\_\_p)
- **`discrete_distribution`** ([`initializer\_list`](#)< `double` > \_\_wl)
- [`result\_type`](#) **`max`** () const
- [`result\_type`](#) **`min`** () const
- template<typename `_UniformRandomNumberGenerator`>  
[`result\_type`](#) **`operator()`** (`_UniformRandomNumberGenerator` &\_\_urng, const [`param\_type`](#) &\_\_p)
- template<typename `_UniformRandomNumberGenerator`>  
[`result\_type`](#) **`operator()`** (`_UniformRandomNumberGenerator` &\_\_urng)
- void [`param`](#) (const [`param\_type`](#) &\_\_param)
- [`param\_type`](#) [`param`](#) () const
- `std::vector`< `double` > [`probabilities`](#) () const
- void **`reset`** ()

### Friends

- template<typename `_IntType1`, typename `_CharT`, typename `_Traits`>  
[`std::basic\_ostream`](#)< `_CharT`, `_Traits` > & **`operator<<`** ([`std::basic\_ostream`](#)< `_CharT`, `_Traits` > &, const [`std::discrete\_distribution`](#)< `_IntType1` > &)
- template<typename `_IntType1`, typename `_CharT`, typename `_Traits`>  
[`std::basic\_istream`](#)< `_CharT`, `_Traits` > & **`operator>>`** ([`std::basic\_istream`](#)< `_CharT`, `_Traits` > &, [`std::discrete\_distribution`](#)< `_IntType1` > &)

### 5.448.1 Detailed Description

`template<typename _IntType = int> class std::discrete_distribution< _IntType >`

A [discrete\\_distribution](#) random number distribution. The formula for the discrete probability mass function is

Definition at line 4659 of file random.h.

### 5.448.2 Member Typedef Documentation

**5.448.2.1** `template<typename _IntType = int> typedef _IntType  
std::discrete_distribution< _IntType >::result_type`

The type of the range of the distribution.

Definition at line 4666 of file random.h.

### 5.448.3 Member Function Documentation

**5.448.3.1** `template<typename _IntType = int> result_type  
std::discrete_distribution< _IntType >::max ( ) const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 4779 of file random.h.

References `std::vector< _Tp, _Alloc >::empty()`, and `std::vector< _Tp, _Alloc >::size()`.

**5.448.3.2** `template<typename _IntType = int> result_type  
std::discrete_distribution< _IntType >::min ( ) const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 4772 of file random.h.

**5.448.3.3** `template<typename _IntType = int> template<typename  
_UniformRandomNumberGenerator > result_type  
std::discrete_distribution< _IntType >::operator() (   
_UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 4790 of file random.h.

References `std::discrete_distribution< _IntType >::operator()()`, and `std::discrete_distribution< _IntType >::param()`.

Referenced by `std::discrete_distribution< _IntType >::operator()()`.

**5.448.3.4** `template<typename _IntType = int> void std::discrete_distribution<  
_IntType >::param ( const param_type & __param ) [inline]`

Sets the parameter set of the distribution.

#### Parameters

`__param` The new parameter set of the distribution.

Definition at line 4765 of file random.h.

**5.448.3.5** `template<typename _IntType = int> param_type  
std::discrete_distribution< _IntType >::param ( ) const  
[inline]`

Returns the parameter set of the distribution.

Definition at line 4757 of file random.h.

Referenced by `std::discrete_distribution< _IntType >::operator()()`, and `std::operator==( )`.

**5.448.3.6** `template<typename _IntType = int> std::vector<double>  
std::discrete_distribution< _IntType >::probabilities ( ) const  
[inline]`

Returns the probabilities of the distribution.

Definition at line 4747 of file random.h.

References `std::vector<_Tp, _Alloc>::empty()`.

**5.448.3.7** `template<typename _IntType = int> void std::discrete_distribution<_IntType>::reset( ) [inline]`

Resets the distribution state.

Definition at line 4740 of file random.h.

#### 5.448.4 Friends And Related Function Documentation

**5.448.4.1** `template<typename _IntType = int> template<typename _IntType1 , typename _CharT , typename _Traits > std::basic_ostream<_CharT, _Traits>& operator<< ( std::basic_ostream<_CharT, _Traits> & , const std::discrete_distribution<_IntType1> & ) [friend]`

Inserts a `discrete_distribution` random number distribution `__x` into the output stream `__os`.

##### Parameters

`__os` An output stream.

`__x` A `discrete_distribution` random number distribution.

##### Returns

The output stream with the state of `__x` inserted or in an error state.

**5.448.4.2** `template<typename _IntType = int> template<typename _IntType1 , typename _CharT , typename _Traits > std::basic_istream<_CharT, _Traits>& operator>> ( std::basic_istream<_CharT, _Traits> & , std::discrete_distribution<_IntType1> & ) [friend]`

Extracts a `discrete_distribution` random number distribution `__x` from the input stream `__is`.

##### Parameters

`__is` An input stream.



## 5.449 `std::discrete_distribution< _IntType >::param_type` Struct Reference

`__x` A `discrete_distribution` random number generator engine.

### Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.449 `std::discrete_distribution< _IntType >::param_type` Struct Reference

### Public Types

- typedef [discrete\\_distribution](#)< `_IntType` > `distribution_type`

### Public Member Functions

- `template<typename _InputIterator >`  
`param_type` (`_InputIterator` \_\_wbegin, `_InputIterator` \_\_wend)
- `template<typename _Func >`  
`param_type` (`size_t` \_\_nw, `double` \_\_xmin, `double` \_\_xmax, `_Func` \_\_fw)
- `param_type` (`const` [param\\_type](#) &)
- `param_type` ([initializer\\_list](#)< `double` > \_\_wil)
- [param\\_type](#) & `operator=` (`const` [param\\_type](#) &)
- [std::vector](#)< `double` > `probabilities` () `const`

### Friends

- `class` `discrete_distribution`< `_IntType` >
- `bool` `operator==` (`const` [param\\_type](#) &\_\_p1, `const` [param\\_type](#) &\_\_p2)

### 5.449.1 Detailed Description

`template<typename _IntType = int> struct std::discrete_distribution< _IntType >::param_type`

Parameter type.

Definition at line 4668 of file `random.h`.

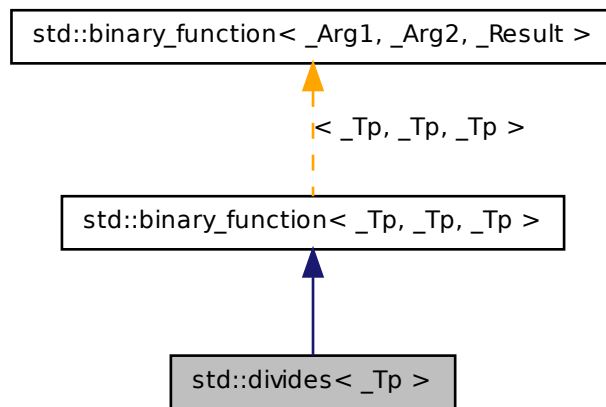
The documentation for this struct was generated from the following files:

- [random.h](#)
- [random.tcc](#)

### 5.450 `std::divides< _Tp >` Struct Template Reference

One of the [math functors](#).

Inheritance diagram for `std::divides< _Tp >`:



#### Public Types

- `typedef _Tp` [first\\_argument\\_type](#)
- `typedef _Tp` [result\\_type](#)
- `typedef _Tp` [second\\_argument\\_type](#)

#### Public Member Functions

- `_Tp operator() (const _Tp &__x, const _Tp &__y) const`

### 5.450.1 Detailed Description

`template<typename _Tp> struct std::divides<_Tp>`

One of the [math functors](#).

Definition at line 168 of file `stl_function.h`.

### 5.450.2 Member Typedef Documentation

**5.450.2.1** `typedef _Tp std::binary_function<_Tp, _Tp, _Tp  
>::first_argument_type [inherited]`

`first_argument_type` is the type of the first argument

Definition at line 118 of file `stl_function.h`.

**5.450.2.2** `typedef _Tp std::binary_function<_Tp, _Tp, _Tp>::result_type  
[inherited]`

`result_type` is the return type

Definition at line 124 of file `stl_function.h`.

**5.450.2.3** `typedef _Tp std::binary_function<_Tp, _Tp, _Tp  
>::second_argument_type [inherited]`

`second_argument_type` is the type of the second argument

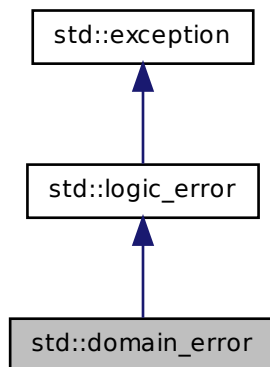
Definition at line 121 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.451 std::domain\_error Class Reference

Inheritance diagram for std::domain\_error:



### Public Member Functions

- **domain\_error** (const [string](#) &\_\_arg)
- virtual const char \* **what** () const throw ()

#### 5.451.1 Detailed Description

Thrown by the library, or by you, to report domain errors (domain in the mathematical sense).

Definition at line 76 of file stdexcept.

#### 5.451.2 Member Function Documentation

##### 5.451.2.1 virtual const char\* std::logic\_error::what ( ) const throw () [virtual, inherited]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future\\_error](#).

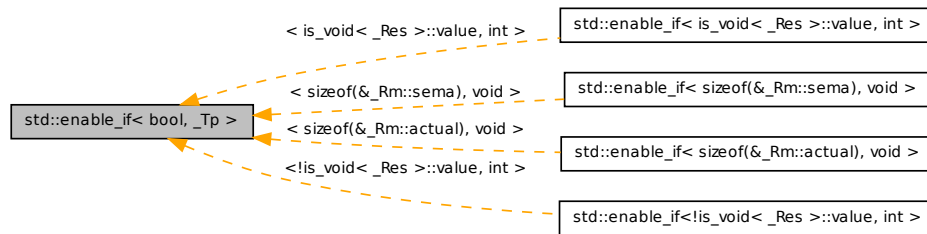
The documentation for this class was generated from the following file:

- [stdexcept](#)

## 5.452 `std::enable_if< bool, _Tp >` Struct Template Reference

[enable\\_if](#)

Inheritance diagram for `std::enable_if< bool, _Tp >`:



### 5.452.1 Detailed Description

`template<bool, typename _Tp = void> struct std::enable_if< bool, _Tp >`

[enable\\_if](#)

Definition at line 836 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.453 `std::enable_shared_from_this< _Tp >` Class Template Reference

Base class allowing use of member function `shared_from_this`.

**Public Member Functions**

- [shared\\_ptr](#)< \_Tp > [shared\\_from\\_this](#) ()
- [shared\\_ptr](#)< const \_Tp > [shared\\_from\\_this](#) () const

**Protected Member Functions**

- [enable\\_shared\\_from\\_this](#) (const [enable\\_shared\\_from\\_this](#) &)
- [enable\\_shared\\_from\\_this](#) & [operator=](#) (const [enable\\_shared\\_from\\_this](#) &)

**Friends**

- `template<typename _Tp1 >`  
`void __enable_shared_from_this_helper` (const \_\_shared\_count<> &\_\_pn,  
const [enable\\_shared\\_from\\_this](#) \*\_\_pe, const \_Tp1 \*\_\_px)

**5.453.1 Detailed Description**

`template<typename _Tp> class std::enable_shared_from_this< _Tp >`

Base class allowing use of member function `shared_from_this`.

Definition at line 474 of file `shared_ptr.h`.

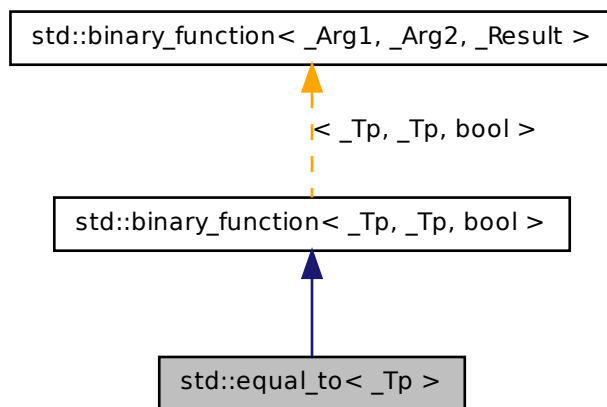
The documentation for this class was generated from the following file:

- [shared\\_ptr.h](#)

**5.454 `std::equal_to< _Tp >` Struct Template Reference**

One of the [comparison functions](#).

Inheritance diagram for std::equal\_to< \_Tp >:



### Public Types

- typedef `_Tp` `first_argument_type`
- typedef `bool` `result_type`
- typedef `_Tp` `second_argument_type`

### Public Member Functions

- `bool operator() (const _Tp &__x, const _Tp &__y) const`

#### 5.454.1 Detailed Description

**template<typename \_Tp> struct std::equal\_to< \_Tp >**

One of the [comparison functors](#).

Definition at line 205 of file `stl_function.h`.

### 5.454.2 Member Typedef Documentation

**5.454.2.1** `typedef _Tp std::binary_function< _Tp , _Tp , bool  
>::first_argument_type [inherited]`

`first_argument_type` is the type of the first argument

Definition at line 118 of file `stl_function.h`.

**5.454.2.2** `typedef bool std::binary_function< _Tp , _Tp , bool >::result_type  
[inherited]`

`result_type` is the return type

Definition at line 124 of file `stl_function.h`.

**5.454.2.3** `typedef _Tp std::binary_function< _Tp , _Tp , bool  
>::second_argument_type [inherited]`

`second_argument_type` is the type of the second argument

Definition at line 121 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.455 `std::error_category` Class Reference

[error\\_category](#)

### Public Member Functions

- `error_category` (const [error\\_category](#) &)
- virtual `error_condition default_error_condition` (int \_\_i) const
- virtual bool `equivalent` (const [error\\_code](#) &\_\_code, int \_\_i) const
- virtual bool `equivalent` (int \_\_i, const [error\\_condition](#) &\_\_cond) const
- virtual `string message` (int) const =0
- virtual const char \* `name` () const =0
- bool `operator!=` (const [error\\_category](#) &\_\_other) const



- `bool operator< (const error\_category &__other) const`
- `error\_category & operator= (const error\_category &)`
- `bool operator== (const error\_category &__other) const`

#### 5.455.1 Detailed Description

##### [error\\_category](#)

Definition at line 66 of file `system_error`.

The documentation for this class was generated from the following file:

- [system\\_error](#)

#### 5.456 `std::error_code` Struct Reference

##### [error\\_code](#)

#### Public Member Functions

- `error_code (int __v, const error\_category &__cat)`
- `template<typename _ErrorCodeEnum >  
error_code (_ErrorCodeEnum __e, typename enable\_if< is\_error\_code\_enum< _ErrorCodeEnum >::value >::type * = 0)`
- `void assign (int __v, const error\_category &__cat)`
- `const error\_category & category () const`
- `void clear ()`
- `error\_condition default_error_condition () const`
- `string message () const`
- `operator bool () const`
- `template<typename _ErrorCodeEnum >  
enable\_if< is\_error\_code\_enum< _ErrorCodeEnum >::value, error\_code & >::type operator= (_ErrorCodeEnum __e)`
- `int value () const`

#### Friends

- `class hash< error\_code >`

### 5.456.1 Detailed Description

[error\\_code](#)

Definition at line 118 of file `system_error`.

The documentation for this struct was generated from the following file:

- [system\\_error](#)

## 5.457 `std::error_condition` Struct Reference

[error\\_condition](#)

### Public Member Functions

- **error\_condition** (int \_\_v, const [error\\_category](#) &\_\_cat)
- template<typename \_ErrorConditionEnum >  
**error\_condition** (\_ErrorConditionEnum \_\_e, typename [enable\\_if](#)< [is\\_error\\_condition\\_enum](#)< \_ErrorConditionEnum >::value >::type !=0)
- void **assign** (int \_\_v, const [error\\_category](#) &\_\_cat)
- const [error\\_category](#) & **category** () const
- void **clear** ()
- [string](#) **message** () const
- **operator bool** () const
- template<typename \_ErrorConditionEnum >  
[enable\\_if](#)< [is\\_error\\_condition\\_enum](#)< \_ErrorConditionEnum >::value, [error\\_condition](#) & >::type **operator=** (\_ErrorConditionEnum \_\_e)
- int **value** () const

### 5.457.1 Detailed Description

[error\\_condition](#)

Definition at line 195 of file `system_error`.

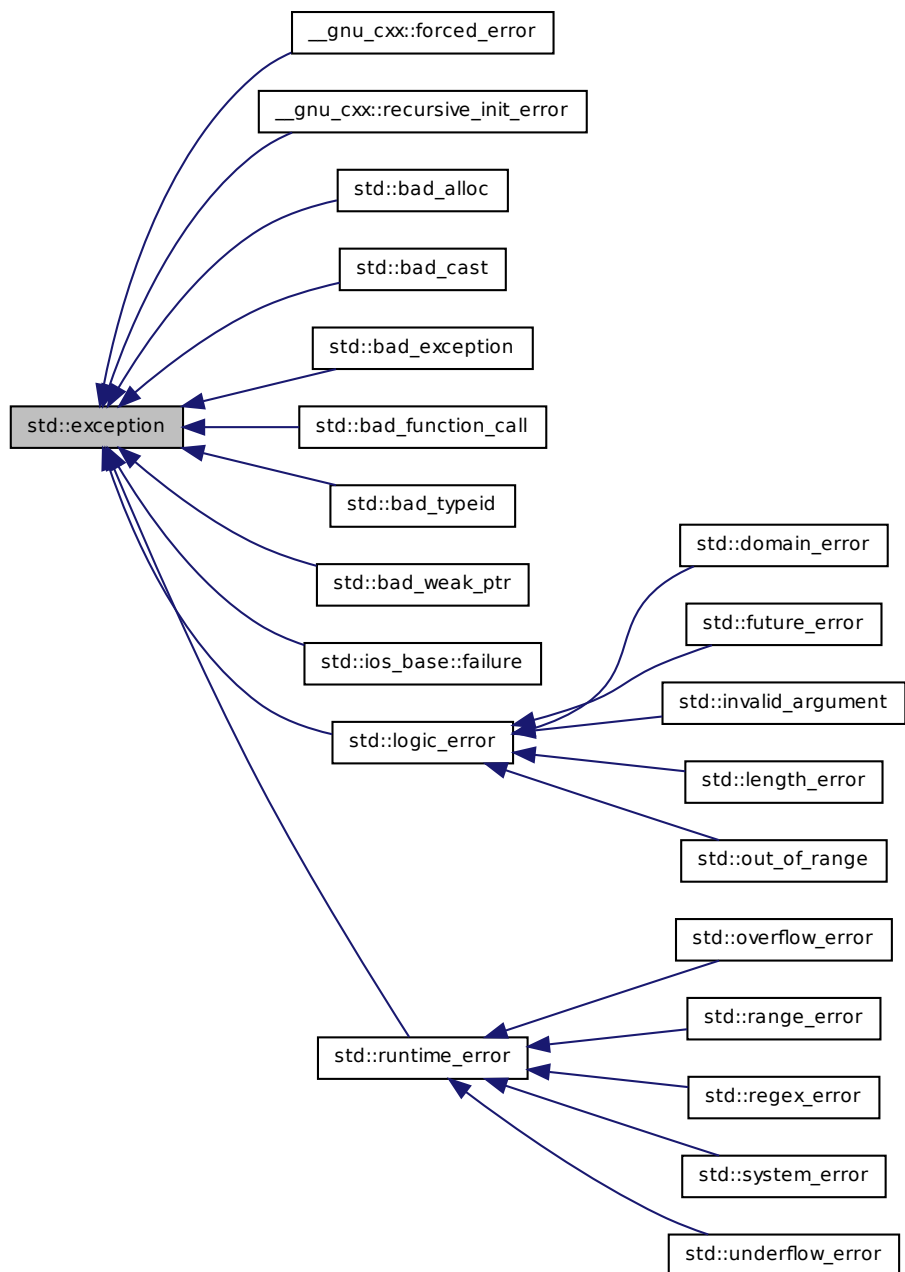
The documentation for this struct was generated from the following file:

- [system\\_error](#)

## 5.458 `std::exception` Class Reference

Base class for all library exceptions.

Inheritance diagram for `std::exception`:



## 5.459 `std::exponential_distribution<_RealType>` Class Template Reference

### Public Member Functions

- virtual const char \* [what](#) () const throw ()

#### 5.458.1 Detailed Description

Base class for all library exceptions. This is the base class for all exceptions thrown by the standard library, and by certain language expressions. You are free to derive your own exception classes, or use a different hierarchy, or to throw non-class data (e.g., fundamental types).

Definition at line 61 of file `exception`.

#### 5.458.2 Member Function Documentation

##### 5.458.2.1 `virtual const char* std::exception::what ( ) const throw ()` [**virtual**]

Returns a C-style character string describing the general cause of the current error.

Reimplemented in `std::bad_exception`, `std::bad_alloc`, `std::bad_cast`, `std::bad_typeid`, `std::future_error`, `std::logic_error`, `std::runtime_error`, `std::ios_base::failure`, and `std::bad_weak_ptr`.

The documentation for this class was generated from the following file:

- [exception](#)

## 5.459 `std::exponential_distribution<_RealType>` Class Template Reference

An exponential continuous distribution for random numbers.

### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_RealType` [result\\_type](#)

### Public Member Functions

- `exponential_distribution` (const `result_type` &\_\_lambda=`result_type`(1))
- `exponential_distribution` (const `param_type` &\_\_p)
- `_RealType lambda` () const
- `result_type max` () const
- `result_type min` () const
- `template<typename _UniformRandomNumberGenerator>`  
`result_type operator()` (\_UniformRandomNumberGenerator &\_\_urng)
- `template<typename _UniformRandomNumberGenerator>`  
`result_type operator()` (\_UniformRandomNumberGenerator &\_\_urng, const `param_type` &\_\_p)
- `param_type param` () const
- `void param` (const `param_type` &\_\_param)
- `void reset` ()

#### 5.459.1 Detailed Description

`template<typename _RealType = double> class std::exponential_distribution<_RealType>`

An exponential continuous distribution for random numbers. The formula for the exponential probability density function is  $p(x|\lambda) = \lambda e^{-\lambda x}$ .

|                    |                         |
|--------------------|-------------------------|
| Mean               | $\frac{1}{\lambda}$     |
| Median             | $\frac{\ln 2}{\lambda}$ |
| Mode               | <i>zero</i>             |
| Range              | $[0, \infty]$           |
| Standard Deviation | $\frac{1}{\lambda^2}$   |

Table 2: Distribution Statistics

Definition at line 4138 of file `random.h`.

#### 5.459.2 Member Typedef Documentation

**5.459.2.1** `template<typename _RealType = double> typedef _RealType std::exponential_distribution<_RealType>::result_type`

The type of the range of the distribution.

Definition at line 4145 of file `random.h`.

### 5.459.3 Constructor & Destructor Documentation

**5.459.3.1** `template<typename _RealType = double> std::exponential_distribution<_RealType>::exponential_distribution ( const result_type & __lambda = result_type (1) ) [inline, explicit]`

Constructs an exponential distribution with inverse scale parameter  $\lambda$ .

Definition at line 4176 of file random.h.

### 5.459.4 Member Function Documentation

**5.459.4.1** `template<typename _RealType = double> _RealType std::exponential_distribution<_RealType>::lambda ( ) const [inline]`

Returns the inverse scale parameter of the distribution.

Definition at line 4197 of file random.h.

**5.459.4.2** `template<typename _RealType = double> result_type std::exponential_distribution<_RealType>::max ( ) const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 4226 of file random.h.

**5.459.4.3** `template<typename _RealType = double> result_type std::exponential_distribution<_RealType>::min ( ) const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 4219 of file random.h.

## 5.459 `std::exponential_distribution<_RealType>` Class Template Reference 2612

**5.459.4.4** `template<typename _RealType = double> template<typename  
_UniformRandomNumberGenerator > result_type  
std::exponential_distribution<_RealType>::operator() (  
_UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 4234 of file random.h.

References `std::exponential_distribution<_RealType>::operator()`, and  
`std::exponential_distribution<_RealType>::param()`.

Referenced by `std::exponential_distribution<_RealType>::operator()`.

**5.459.4.5** `template<typename _RealType = double> param_type  
std::exponential_distribution<_RealType>::param ( ) const  
[inline]`

Returns the parameter set of the distribution.

Definition at line 4204 of file random.h.

Referenced by `std::exponential_distribution<_RealType>::operator()`,  
`std::operator==( )`, and `std::operator>>()`.

**5.459.4.6** `template<typename _RealType = double> void  
std::exponential_distribution<_RealType>::param ( const  
param_type & __param ) [inline]`

Sets the parameter set of the distribution.

### Parameters

**`__param`** The new parameter set of the distribution.

Definition at line 4212 of file random.h.

**5.459.4.7** `template<typename _RealType = double> void  
std::exponential_distribution<_RealType>::reset ( ) [inline]`

Resets the distribution state.

Has no effect on exponential distributions.

Definition at line 4191 of file `random.h`.

The documentation for this class was generated from the following file:

- [random.h](#)

## **5.460** `std::exponential_distribution<_RealType>::param_type` Struct Reference

### Public Types

- typedef `exponential_distribution<_RealType>` `distribution_type`

### Public Member Functions

- `param_type` (`_RealType` \_\_lambda=`_RealType`(1))
- `_RealType` `lambda` () const

### Friends

- bool `operator==` (const `param_type` &\_\_p1, const `param_type` &\_\_p2)

#### **5.460.1** Detailed Description

`template<typename _RealType = double> struct std::exponential_distribution<_RealType>::param_type`

Parameter type.

Definition at line 4147 of file `random.h`.

The documentation for this struct was generated from the following file:

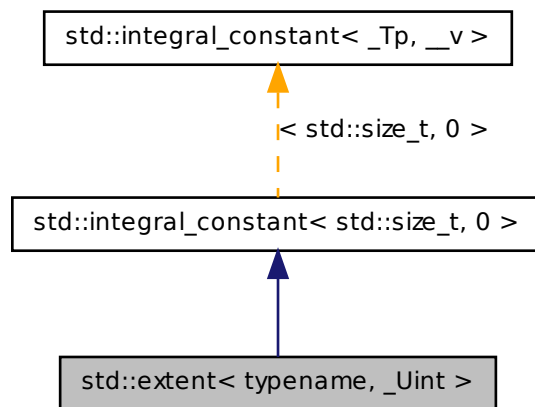
- [random.h](#)

## **5.461** `std::extent<typename, _Uint>` Struct Template Reference

`extent`



Inheritance diagram for std::extent< typename, \_Uint >:



### Public Types

- typedef `integral_constant< std::size_t, __v >` **type**
- typedef `std::size_t` **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr `std::size_t` **value**

#### 5.461.1 Detailed Description

**template<typename, unsigned \_Uint = 0> struct std::extent< typename, \_Uint >**

extent

Definition at line 385 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## **5.462 std::extreme\_value\_distribution< \_RealType > Class Template Reference**

A [extreme\\_value\\_distribution](#) random number distribution.

### **Classes**

- struct [param\\_type](#)

### **Public Types**

- typedef \_RealType [result\\_type](#)

### **Public Member Functions**

- **extreme\_value\_distribution** (\_RealType \_\_a=\_RealType(0), \_RealType \_\_b=\_RealType(1))
- **extreme\_value\_distribution** (const [param\\_type](#) &\_\_p)
- \_RealType **a** () const
- \_RealType **b** () const
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- void **param** (const [param\\_type](#) &\_\_param)
- [param\\_type](#) **param** () const
- void **reset** ()

### 5.462.1 Detailed Description

```
template<typename _RealType = double> class std::extreme_value_
distribution<_RealType>
```

A [extreme\\_value\\_distribution](#) random number distribution. The formula for the normal probability mass function is

$$p(x|a,b) = \frac{1}{b} \exp\left(\frac{a-x}{b} - \exp\left(\frac{a-x}{b}\right)\right)$$

Definition at line 4487 of file random.h.

### 5.462.2 Member Typedef Documentation

**5.462.2.1** `template<typename _RealType = double> typedef _RealType  
std::extreme_value_distribution<_RealType>::result_type`

The type of the range of the distribution.

Definition at line 4494 of file random.h.

### 5.462.3 Member Function Documentation

**5.462.3.1** `template<typename _RealType = double> _RealType  
std::extreme_value_distribution<_RealType>::a ( ) const  
[inline]`

Return the  $a$  parameter of the distribution.

Definition at line 4545 of file random.h.

**5.462.3.2** `template<typename _RealType = double> _RealType  
std::extreme_value_distribution<_RealType>::b ( ) const  
[inline]`

Return the  $b$  parameter of the distribution.

Definition at line 4552 of file random.h.

**5.462.3.3** `template<typename _RealType = double> result_type  
std::extreme_value_distribution<_RealType>::max ( ) const  
[inline]`

Returns the least upper bound value of the distribution.

Definition at line 4581 of file random.h.

**5.462.3.4** `template<typename _RealType = double> result_type  
std::extreme_value_distribution<_RealType>::min ( ) const  
[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 4574 of file random.h.

**5.462.3.5** `template<typename _RealType = double> template<typename  
_UniformRandomNumberGenerator> result_type  
std::extreme_value_distribution<_RealType>::operator() (   
_UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 4589 of file random.h.

References `std::extreme_value_distribution<_RealType>::operator()()`, and  
`std::extreme_value_distribution<_RealType>::param()`.

Referenced by `std::extreme_value_distribution<_RealType>::operator()()`.

**5.462.3.6** `template<typename _RealType = double> void  
std::extreme_value_distribution<_RealType>::param ( const  
param_type & __param ) [inline]`

Sets the parameter set of the distribution.

#### Parameters

`__param` The new parameter set of the distribution.

Definition at line 4567 of file random.h.

**5.462.3.7** `template<typename _RealType = double> param_type  
std::extreme_value_distribution< _RealType >::param ( ) const  
[inline]`

Returns the parameter set of the distribution.

Definition at line 4559 of file `random.h`.

Referenced by `std::extreme_value_distribution< _RealType >::operator()()`, `std::operator==( )`, and `std::operator>>()`.

**5.462.3.8** `template<typename _RealType = double> void  
std::extreme_value_distribution< _RealType >::reset ( )  
[inline]`

Resets the distribution state.

Definition at line 4538 of file `random.h`.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.463 `std::extreme_value_distribution< _RealType >::param_type` Struct Reference

### Public Types

- typedef [extreme\\_value\\_distribution< \\_RealType >](#) `distribution_type`

### Public Member Functions

- `param_type` (`_RealType __a=_RealType(0), _RealType __b=_RealType(1)`)
- `_RealType a` () const
- `_RealType b` () const

### Friends

- bool `operator==(const param\_type &__p1, const param\_type &__p2)`

### 5.463.1 Detailed Description

```
template<typename _RealType = double> struct std::extreme_value_
distribution<_RealType>::param_type
```

Parameter type.

Definition at line 4496 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.464 `std::fisher_f_distribution<_RealType>` Class Template Reference

A [fisher\\_f\\_distribution](#) random number distribution.

### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_RealType` [result\\_type](#)

### Public Member Functions

- **fisher\_f\_distribution** (`_RealType __m=_RealType(1), _RealType __n=_RealType(1)`)
- **fisher\_f\_distribution** (const [param\\_type](#) &\_\_p)
- `_RealType m` () const
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- `_RealType n` () const
- template<typename `_UniformRandomNumberGenerator`>  
[result\\_type](#) **operator()** (`_UniformRandomNumberGenerator &__urng, const param\_type &__p`)
- template<typename `_UniformRandomNumberGenerator`>  
[result\\_type](#) **operator()** (`_UniformRandomNumberGenerator &__urng`)
- void **param** (const [param\\_type](#) &\_\_param)
- [param\\_type](#) **param** () const
- void **reset** ()

## Friends

- `template<typename _RealType1, typename _CharT, typename _Traits>`  
`std::basic_ostream<_CharT, _Traits> & operator<< (std::basic_ostream<_CharT, _Traits> &, const std::fisher_f_distribution<_RealType1> &)`
- `template<typename _RealType1>`  
`bool operator== (const std::fisher_f_distribution<_RealType1> &__d1, const std::fisher_f_distribution<_RealType1> &__d2)`
- `template<typename _RealType1, typename _CharT, typename _Traits>`  
`std::basic_istream<_CharT, _Traits> & operator>> (std::basic_istream<_CharT, _Traits> &, std::fisher_f_distribution<_RealType1> &)`

### 5.464.1 Detailed Description

`template<typename _RealType = double> class std::fisher_f_distribution<_RealType>`

A [fisher\\_f\\_distribution](#) random number distribution. The formula for the normal probability mass function is

$$p(x|m, n) = \frac{\Gamma((m+n)/2)}{\Gamma(m/2)\Gamma(n/2)} \left(\frac{m}{n}\right)^{m/2} x^{(m/2)-1} \left(1 + \frac{mx}{n}\right)^{-(m+n)/2}$$

Definition at line 2854 of file `random.h`.

### 5.464.2 Member Typedef Documentation

**5.464.2.1** `template<typename _RealType = double> typedef _RealType std::fisher_f_distribution<_RealType>::result_type`

The type of the range of the distribution.

Definition at line 2861 of file `random.h`.

### 5.464.3 Member Function Documentation

**5.464.3.1** `template<typename _RealType = double> result_type std::fisher_f_distribution<_RealType>::max( ) const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 2948 of file `random.h`.

**5.464.3.2** `template<typename _RealType = double> result_type  
std::fisher_f_distribution<_RealType>::min ( ) const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 2941 of file random.h.

**5.464.3.3** `template<typename _RealType = double> template<typename  
_UniformRandomNumberGenerator > result_type  
std::fisher_f_distribution<_RealType>::operator() (   
_UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 2956 of file random.h.

**5.464.3.4** `template<typename _RealType = double> void  
std::fisher_f_distribution<_RealType>::param ( const param_type  
& __param ) [inline]`

Sets the parameter set of the distribution.

#### Parameters

`__param` The new parameter set of the distribution.

Definition at line 2934 of file random.h.

**5.464.3.5** `template<typename _RealType = double> param_type  
std::fisher_f_distribution<_RealType>::param ( ) const  
[inline]`

Returns the parameter set of the distribution.

Definition at line 2926 of file random.h.



**5.464.3.6** `template<typename _RealType = double> void  
std::fisher_f_distribution<_RealType>::reset( ) [inline]`

Resets the distribution state.

Definition at line 2905 of file random.h.

References `std::gamma_distribution<_RealType>::reset()`.

#### 5.464.4 Friends And Related Function Documentation

**5.464.4.1** `template<typename _RealType = double> template<typename  
_RealType1 , typename _CharT , typename _Traits >  
std::basic_ostream<_CharT, _Traits>& operator<<  
( std::basic_ostream<_CharT, _Traits> & , const  
std::fisher_f_distribution<_RealType1> & ) [friend]`

Inserts a `fisher_f_distribution` random number distribution `__x` into the output stream `__os`.

##### Parameters

`__os` An output stream.

`__x` A `fisher_f_distribution` random number distribution.

##### Returns

The output stream with the state of `__x` inserted or in an error state.

**5.464.4.2** `template<typename _RealType = double> template<typename  
_RealType1 > bool operator==( const std::fisher_f_distribution<  
_RealType1 > & __d1, const std::fisher_f_distribution<_RealType1  
> & __d2 ) [friend]`

Return true if two Fisher f distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 2977 of file random.h.

## 5.465 `std::fisher_f_distribution<_RealType>::param_type` Struct Reference 2623

5.464.4.3 `template<typename _RealType = double> template<typename  
_RealType1, typename _CharT, typename _Traits >  
std::basic_istream<_CharT, _Traits>& operator>>  
( std::basic_istream<_CharT, _Traits> & ,  
std::fisher_f_distribution<_RealType1> & ) [friend]`

Extracts a `fisher_f_distribution` random number distribution `__x` from the input stream `__is`.

### Parameters

- `__is` An input stream.
- `__x` A `fisher_f_distribution` random number generator engine.

### Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

## 5.465 `std::fisher_f_distribution<_RealType>::param_type` Struct Reference

### Public Types

- typedef `fisher_f_distribution<_RealType>` `distribution_type`

### Public Member Functions

- `param_type` (`_RealType __m=_RealType(1), _RealType __n=_RealType(1)`)
- `_RealType m` () const
- `_RealType n` () const

### Friends

- bool `operator==` (const `param_type` &\_\_p1, const `param_type` &\_\_p2)

### 5.465.1 Detailed Description

```
template<typename _RealType = double> struct std::fisher_f_distribution< _
_RealType >::param_type
```

Parameter type.

Definition at line 2863 of file random.h.

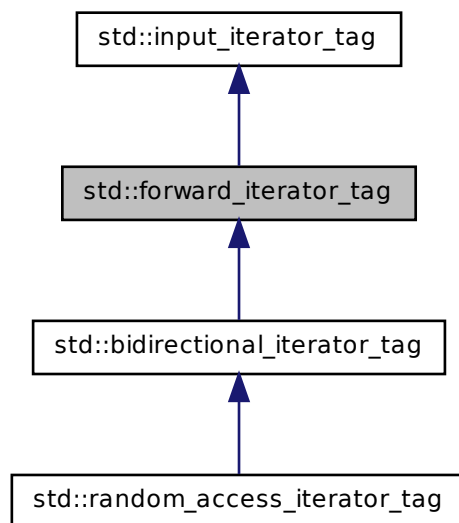
The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.466 std::forward\_iterator\_tag Struct Reference

Forward iterators support a superset of input iterator operations.

Inheritance diagram for std::forward\_iterator\_tag:



### 5.466.1 Detailed Description

Forward iterators support a superset of input iterator operations.

Definition at line 96 of file `stl_iterator_base_types.h`.

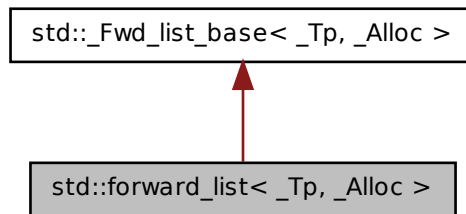
The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 5.467 `std::forward_list< _Tp, _Alloc >` Class Template Reference

A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.

Inheritance diagram for `std::forward_list< _Tp, _Alloc >`:



### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_Fwd_list_const_iterator< _Tp >` **const\_iterator**
- typedef `_Tp_alloc_type::const_pointer` **const\_pointer**
- typedef `_Tp_alloc_type::const_reference` **const\_reference**
- typedef `std::ptrdiff_t` **difference\_type**
- typedef `_Fwd_list_iterator< _Tp >` **iterator**
- typedef `_Tp_alloc_type::pointer` **pointer**
- typedef `_Tp_alloc_type::reference` **reference**
- typedef `std::size_t` **size\_type**
- typedef `_Tp` **value\_type**

**Public Member Functions**

- [forward\\_list](#) (const \_Alloc &\_\_al=\_Alloc())
- [forward\\_list](#) (const [forward\\_list](#) &\_\_list, const \_Alloc &\_\_al)
- [forward\\_list](#) (size\_type \_\_n)
- [forward\\_list](#) ([forward\\_list](#) &&\_\_list)
- [forward\\_list](#) (std::initializer\_list< \_Tp > \_\_il, const \_Alloc &\_\_al=\_Alloc())
- [forward\\_list](#) (size\_type \_\_n, const \_Tp &\_\_value, const \_Alloc &\_\_al=\_Alloc())
- [forward\\_list](#) ([forward\\_list](#) &&\_\_list, const \_Alloc &\_\_al)
- template<typename \_InputIterator >  
[forward\\_list](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Alloc &\_\_al=\_Alloc())
- [forward\\_list](#) (const [forward\\_list](#) &\_\_list)
- [~forward\\_list](#) ()
- void [assign](#) (std::initializer\_list< \_Tp > \_\_il)
- template<typename \_InputIterator >  
void [assign](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void [assign](#) (size\_type \_\_n, const \_Tp &\_\_val)
- [iterator before\\_begin](#) ()
- [const\\_iterator before\\_begin](#) () const
- [iterator begin](#) ()
- [const\\_iterator begin](#) () const
- [const\\_iterator cbefore\\_begin](#) () const
- [const\\_iterator cbegin](#) () const
- [const\\_iterator cend](#) () const
- void [clear](#) ()
- template<typename... \_Args>  
[iterator emplace\\_after](#) (const\_iterator \_\_pos, \_Args &&...\_\_args)
- template<typename... \_Args>  
void [emplace\\_front](#) (\_Args &&...\_\_args)
- bool [empty](#) () const
- [const\\_iterator end](#) () const
- [iterator end](#) ()
- [iterator erase\\_after](#) (const\_iterator \_\_pos)
- [iterator erase\\_after](#) (const\_iterator \_\_pos, const\_iterator \_\_last)
- reference [front](#) ()
- const\_reference [front](#) () const
- allocator\_type [get\\_allocator](#) () const
- template<typename \_InputIterator >  
[iterator insert\\_after](#) (const\_iterator \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last)
- [iterator insert\\_after](#) (const\_iterator \_\_pos, const \_Tp &\_\_val)
- [iterator insert\\_after](#) (const\_iterator \_\_pos, \_Tp &&\_\_val)

- `iterator insert_after` (`const_iterator` \_\_pos, `size_type` \_\_n, `const _Tp` &\_\_val)
- `iterator insert_after` (`const_iterator` \_\_pos, `std::initializer_list`< `_Tp` > \_\_il)
- `size_type max_size` () `const`
- `template`<`typename` \_Comp >  
void `merge` (`forward_list` && \_\_list, \_Comp \_\_comp)
- void `merge` (`forward_list` && \_\_list)
- `forward_list` & `operator=` (`std::initializer_list`< `_Tp` > \_\_il)
- `forward_list` & `operator=` (`const forward_list` & \_\_list)
- `forward_list` & `operator=` (`forward_list` && \_\_list)
- void `pop_front` ()
- void `push_front` (`const _Tp` &\_\_val)
- void `push_front` (`_Tp` && \_\_val)
- void `remove` (`const _Tp` &\_\_val)
- `template`<`typename` \_Pred >  
void `remove_if` (\_Pred \_\_pred)
- void `resize` (`size_type` \_\_sz)
- void `resize` (`size_type` \_\_sz, `const value_type` &\_\_val)
- void `reverse` ()
- `template`<`typename` \_Comp >  
void `sort` (\_Comp \_\_comp)
- void `sort` ()
- void `splice_after` (`const_iterator` \_\_pos, `forward_list` && \_\_list, `const_iterator` \_\_before, `const_iterator` \_\_last)
- void `splice_after` (`const_iterator` \_\_pos, `forward_list` && \_\_list, `const_iterator` \_\_i)
- void `splice_after` (`const_iterator` \_\_pos, `forward_list` && \_\_list)
- void `swap` (`forward_list` & \_\_list)
- `template`<`typename` \_BinPred >  
void `unique` (\_BinPred \_\_binary\_pred)
- void `unique` ()

### Private Types

- `typedef` \_Alloc::template `rebind`< `_Fwd_list_node`< `_Tp` > ::other `_Node_alloc_type`

### Private Member Functions

- `template`<`typename`... \_Args>  
`_Node` \* `_M_create_node` (\_Args &&... \_\_args)
- `_Fwd_list_node_base` \* `_M_erase_after` (`_Fwd_list_node_base` \* \_\_pos)
- `_Fwd_list_node_base` \* `_M_erase_after` (`_Fwd_list_node_base` \* \_\_pos, `_Fwd_list_node_base` \* \_\_last)

- `_Node * _M_get_node ()`
- `const _Node_alloc_type & _M_get_Node_allocator () const`
- `_Node_alloc_type & _M_get_Node_allocator ()`
- `template<typename... _Args>  
_Fwd_list_node_base * _M_insert_after (const_iterator __pos, _Args &&... _args)`
- `void _M_put_node (_Node * __p)`

#### Private Attributes

- `_Fwd_list_impl _M_impl`

#### 5.467.1 Detailed Description

`template<typename _Tp, typename _Alloc = allocator<_Tp>> class std::forward_list<_Tp, _Alloc>`

A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence. Meets the requirements of a [container](#), a [sequence](#), including the [optional sequence requirements](#) with the exception of `at` and `operator[]`.

This is a *singly linked* list. Traversal up the list requires linear time, but adding and removing elements (or *nodes*) is done in constant time, regardless of where the change takes place. Unlike `std::vector` and `std::deque`, random-access iterators are not provided, so subscripting ( `[]` ) access is not allowed. For algorithms which only need sequential access, this lack makes no difference.

Also unlike the other standard containers, `std::forward_list` provides specialized algorithms unique to linked lists, such as splicing, sorting, and in-place reversal.

A couple points on memory allocation for `forward_list<Tp>`:

First, we never actually allocate a `Tp`, we allocate `Fwd_list_node<Tp>`'s and trust [20.1.5]/4 to DTRT. This is to ensure that after elements from `forward_list<X, Alloc1>` are spliced into `forward_list<X, Alloc2>`, destroying the memory of the second list is a valid operation, i.e., `Alloc1` giveth and `Alloc2` taketh away.

Definition at line 408 of file `forward_list.h`.

## 5.467.2 Constructor & Destructor Documentation

**5.467.2.1** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
std::forward_list<_Tp, _Alloc>::forward_list ( const _Alloc & __al  
= _Alloc() ) [inline, explicit]`

Creates a forward\_list with no elements.

### Parameters

*al* An allocator object.

Definition at line 437 of file forward\_list.h.

**5.467.2.2** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
std::forward_list<_Tp, _Alloc>::forward_list ( const forward_list<  
_Tp, _Alloc> & __list, const _Alloc & __al ) [inline]`

Copy constructor with allocator argument.

### Parameters

*list* Input list to copy.

*al* An allocator object.

Definition at line 446 of file forward\_list.h.

**5.467.2.3** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
std::forward_list<_Tp, _Alloc>::forward_list ( forward_list<_Tp,  
_Alloc> && __list, const _Alloc & __al ) [inline]`

Move constructor with allocator argument.

### Parameters

*list* Input list to move.

*al* An allocator object.

Definition at line 455 of file forward\_list.h.



**5.467.2.4** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
std::forward_list<_Tp, _Alloc>::forward_list ( size_type __n )  
[inline, explicit]`

Creates a forward\_list with default constructed elements.

#### Parameters

*n* The number of elements to initially create.

This constructor creates the forward\_list with *n* default constructed elements.

Definition at line 467 of file forward\_list.h.

**5.467.2.5** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
std::forward_list<_Tp, _Alloc>::forward_list ( size_type __n,  
const _Tp & __value, const _Alloc & __al = _Alloc() )  
[inline]`

Creates a forward\_list with copies of an exemplar element.

#### Parameters

*n* The number of elements to initially create.

*value* An element to copy.

*al* An allocator object.

This constructor fills the forward\_list with *n* copies of *value*.

Definition at line 480 of file forward\_list.h.

**5.467.2.6** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
template<typename _InputIterator> std::forward_list<_Tp, _Alloc>  
>::forward_list ( _InputIterator __first, _InputIterator __last,  
const _Alloc & __al = _Alloc() ) [inline]`

Builds a forward\_list from a range.

#### Parameters

*first* An input iterator.

*last* An input iterator.

*al* An allocator object.

Create a forward\_list consisting of copies of the elements from [*first*,*last*). This is linear in N (where N is distance(*first*,*last*)).

Definition at line 496 of file forward\_list.h.

```
5.467.2.7 template<typename _Tp, typename _Alloc = allocator<_Tp>>
 std::forward_list< _Tp, _Alloc >::forward_list (const forward_list<
 _Tp, _Alloc > & __list) [inline]
```

The forward\_list copy constructor.

#### Parameters

*list* A forward\_list of identical element and allocator types.

The newly-created forward\_list uses a copy of the allocation object used by *list*.

Definition at line 513 of file forward\_list.h.

References std::forward\_list< \_Tp, \_Alloc >::begin(), and std::forward\_list< \_Tp, \_Alloc >::end().

```
5.467.2.8 template<typename _Tp, typename _Alloc = allocator<_Tp>>
 std::forward_list< _Tp, _Alloc >::forward_list (forward_list< _Tp,
 _Alloc > && __list) [inline]
```

The forward\_list move constructor.

#### Parameters

*list* A forward\_list of identical element and allocator types.

The newly-created forward\_list contains the exact contents of [forward\\_list](#). The contents of *list* are a valid, but unspecified forward\_list.

Definition at line 526 of file forward\_list.h.

**5.467.2.9** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
std::forward_list<_Tp, _Alloc>::forward_list ( std::initializer_list<  
_Tp> __il, const _Alloc & __al = _Alloc() ) [inline]`

Builds a forward\_list from an [initializer\\_list](#).

#### Parameters

*il* An [initializer\\_list](#) of value\_type.

*al* An allocator object.

Create a forward\_list consisting of copies of the elements in the [initializer\\_list](#) *il*. This is linear in *il.size()*.

Definition at line 537 of file forward\_list.h.

**5.467.2.10** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
std::forward_list<_Tp, _Alloc>::~~forward_list ( ) [inline]`

The [forward\\_list](#) dtor.

Definition at line 545 of file forward\_list.h.

### 5.467.3 Member Function Documentation

**5.467.3.1** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
template<typename _InputIterator> void std::forward_list<_Tp,  
_Alloc>::assign ( _InputIterator __first, _InputIterator __last )  
[inline]`

Assigns a range to a forward\_list.

#### Parameters

*first* An input iterator.

*last* An input iterator.

This function fills a forward\_list with copies of the elements in the range [*first*,*last*).

Note that the assignment completely changes the forward\_list and that the resulting forward\_list's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 607 of file `forward_list.h`.

References `std::forward_list<_Tp, _Alloc>::cbefore_begin()`, `std::forward_list<_Tp, _Alloc>::clear()`, and `std::forward_list<_Tp, _Alloc>::insert_after()`.

Referenced by `std::forward_list<_Tp, _Alloc>::operator=()`.

**5.467.3.2** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void  
std::forward_list<_Tp, _Alloc>::assign ( size_type __n, const_Tp  
& __val ) [inline]`

Assigns a given value to a `forward_list`.

#### Parameters

*n* Number of elements to be assigned.

*val* Value to be assigned.

This function fills a `forward_list` with *n* copies of the given value. Note that the assignment completely changes the `forward_list` and that the resulting `forward_list`'s size is the same as the number of elements assigned. Old data may be lost.

Definition at line 624 of file `forward_list.h`.

References `std::forward_list<_Tp, _Alloc>::cbefore_begin()`, `std::forward_list<_Tp, _Alloc>::clear()`, and `std::forward_list<_Tp, _Alloc>::insert_after()`.

**5.467.3.3** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void  
std::forward_list<_Tp, _Alloc>::assign ( std::initializer_list<_Tp  
> __il ) [inline]`

Assigns an [initializer\\_list](#) to a `forward_list`.

#### Parameters

*il* An [initializer\\_list](#) of `value_type`.

Replace the contents of the `forward_list` with copies of the elements in the [initializer\\_list](#) *il*. This is linear in `il.size()`.

Definition at line 639 of file `forward_list.h`.

References `std::forward_list<_Tp, _Alloc>::cbefore_begin()`, `std::forward_list<_Tp, _Alloc>::clear()`, and `std::forward_list<_Tp, _Alloc>::insert_after()`.

**5.467.3.4** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
iterator std::forward_list< _Tp, _Alloc >::before_begin ( )  
[inline]`

Returns a read/write iterator that points before the first element in the forward\_list. Iteration is done in ordinary element order.

Definition at line 657 of file forward\_list.h.

Referenced by std::forward\_list< \_Tp, \_Alloc >::operator=(), and std::forward\_list< \_Tp, \_Alloc >::resize().

**5.467.3.5** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
const_iterator std::forward_list< _Tp, _Alloc >::before_begin ( )  
const [inline]`

Returns a read-only (constant) iterator that points before the first element in the forward\_list. Iteration is done in ordinary element order.

Definition at line 666 of file forward\_list.h.

**5.467.3.6** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
iterator std::forward_list< _Tp, _Alloc >::begin ( ) [inline]`

Returns a read/write iterator that points to the first element in the forward\_list. Iteration is done in ordinary element order.

Definition at line 674 of file forward\_list.h.

Referenced by std::forward\_list< \_Tp, \_Alloc >::forward\_list(), std::forward\_list< \_Tp, \_Alloc >::operator=(), and std::forward\_list< \_Tp, \_Alloc >::unique().

**5.467.3.7** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
const_iterator std::forward_list< _Tp, _Alloc >::begin ( ) const  
[inline]`

Returns a read-only (constant) iterator that points to the first element in the forward\_list. Iteration is done in ordinary element order.

Definition at line 683 of file forward\_list.h.

**5.467.3.8** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
const_iterator std::forward_list< _Tp, _Alloc >::cbefore_begin ( )  
const [inline]`

Returns a read-only (constant) iterator that points before the first element in the forward\_list. Iteration is done in ordinary element order.

Definition at line 719 of file forward\_list.h.

Referenced by std::forward\_list< \_Tp, \_Alloc >::assign(), std::forward\_list< \_Tp, \_Alloc >::emplace\_front(), and std::forward\_list< \_Tp, \_Alloc >::push\_front().

**5.467.3.9** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
const_iterator std::forward_list< _Tp, _Alloc >::cbegin ( ) const  
[inline]`

Returns a read-only (constant) iterator that points to the first element in the forward\_list. Iteration is done in ordinary element order.

Definition at line 710 of file forward\_list.h.

Referenced by std::forward\_list< \_Tp, \_Alloc >::operator=(), and std::operator==( ).

**5.467.3.10** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
const_iterator std::forward_list< _Tp, _Alloc >::cend ( ) const  
[inline]`

Returns a read-only (constant) iterator that points one past the last element in the forward\_list. Iteration is done in ordinary element order.

Definition at line 728 of file forward\_list.h.

Referenced by std::forward\_list< \_Tp, \_Alloc >::operator=(), and std::operator==( ).

**5.467.3.11** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void  
std::forward_list< _Tp, _Alloc >::clear ( ) [inline]`

Erases all the elements.

Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1024 of file forward\_list.h.

Referenced by std::forward\_list< \_Tp, \_Alloc >::assign(), and std::forward\_list< \_Tp, \_Alloc >::operator=().

**5.467.3.12** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
template<typename... _Args> iterator std::forward_list< _Tp,  
_Alloc >::emplace_after ( const_iterator __pos, _Args &&...  
__args ) [inline]`

Constructs object in forward\_list after the specified iterator.

#### Parameters

*pos* A const\_iterator into the forward\_list.

*args* Arguments.

#### Returns

An iterator that points to the inserted data.

This function will insert an object of type T constructed with T(std::forward<Args>(args)...) after the specified location. Due to the nature of a forward\_list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 841 of file forward\_list.h.

**5.467.3.13** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
template<typename... _Args> void std::forward_list< _Tp, _Alloc  
>::emplace_front ( _Args &&... __args ) [inline]`

Constructs object in forward\_list at the front of the list.

#### Parameters

*args* Arguments.

This function will insert an object of type Tp constructed with Tp(std::forward<Args>(args)...) at the front of the list. Due to the nature of a forward\_list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 785 of file forward\_list.h.

References std::forward\_list< \_Tp, \_Alloc >::cbefore\_begin().

**5.467.3.14** `template<typename _Tp, typename _Alloc = allocator<_Tp>> bool  
std::forward_list< _Tp, _Alloc >::empty ( ) const [inline]`

Returns true if the forward\_list is empty. (Thus `begin()` would equal `end()`.)

Definition at line 736 of file forward\_list.h.

Referenced by `std::forward_list< _Tp, _Alloc >::insert_after()`.

**5.467.3.15** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
iterator std::forward_list< _Tp, _Alloc >::end ( ) [inline]`

Returns a read/write iterator that points one past the last element in the forward\_list. Iteration is done in ordinary element order.

Definition at line 692 of file forward\_list.h.

Referenced by `std::forward_list< _Tp, _Alloc >::forward_list()`, `std::forward_list< _Tp, _Alloc >::operator=()`, `std::forward_list< _Tp, _Alloc >::resize()`, and `std::forward_list< _Tp, _Alloc >::unique()`.

**5.467.3.16** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
const_iterator std::forward_list< _Tp, _Alloc >::end ( ) const  
[inline]`

Returns a read-only iterator that points one past the last element in the forward\_list. Iteration is done in ordinary element order.

Definition at line 701 of file forward\_list.h.

**5.467.3.17** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
iterator std::forward_list< _Tp, _Alloc >::erase_after (   
const_iterator __pos, const_iterator __last ) [inline]`

Remove a range of elements.

#### Parameters

*pos* Iterator pointing before the first element to be erased.

*last* Iterator pointing to one past the last element to be erased.

#### Returns

.



This function will erase the elements in the range (pos,last) and shorten the forward\_list accordingly.

This operation is linear time in the size of the range and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 965 of file forward\_list.h.

**5.467.3.18** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
iterator std::forward_list< _Tp, _Alloc >::erase_after (   
const_iterator __pos ) [inline]`

Removes the element pointed to by the iterator following pos.

#### Parameters

*pos* Iterator pointing before element to be erased.

#### Returns

An iterator pointing to the element following the one that was erased, or `end()` if no such element exists.

This function will erase the element at the given position and thus shorten the forward\_list by one.

Due to the nature of a forward\_list this operation can be done in constant time, and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 942 of file forward\_list.h.

Referenced by `std::forward_list< _Tp, _Alloc >::operator=()`, `std::forward_list< _Tp, _Alloc >::resize()`, and `std::forward_list< _Tp, _Alloc >::unique()`.

**5.467.3.19** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
reference std::forward_list< _Tp, _Alloc >::front ( ) [inline]`

Returns a read/write reference to the data at the first element of the forward\_list.

Definition at line 753 of file forward\_list.h.

**5.467.3.20** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
const_reference std::forward_list< _Tp, _Alloc >::front ( ) const  
[inline]`

Returns a read-only (constant) reference to the data at the first element of the forward\_list.

Definition at line 764 of file forward\_list.h.

**5.467.3.21** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
allocator_type std::forward_list< _Tp, _Alloc >::get_allocator ( )  
const [inline]`

Get a copy of the memory allocation object.

Definition at line 647 of file forward\_list.h.

**5.467.3.22** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
iterator std::forward_list< _Tp, _Alloc >::insert_after (   
const_iterator __pos, const _Tp & __val ) [inline]`

Inserts given value into forward\_list after specified iterator.

#### Parameters

*pos* An iterator into the forward\_list.

*val* Data to be inserted.

#### Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value after the specified location. Due to the nature of a forward\_list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 858 of file forward\_list.h.

Referenced by std::forward\_list< \_Tp, \_Alloc >::assign(), std::forward\_list< \_Tp, \_Alloc >::operator=(), and std::forward\_list< \_Tp, \_Alloc >::resize().

**5.467.3.23** `template<typename _Tp, typename _Alloc > forward_list< _Tp,  
_Alloc >::iterator forward_list::insert_after ( const_iterator __pos,  
size_type __n, const _Tp & __val )`

Inserts a number of copies of given data into the forward\_list.

#### Parameters

*pos* An iterator into the forward\_list.

*n* Number of elements to be inserted.

*val* Data to be inserted.

#### Returns

An iterator pointing to the last inserted copy of *val* or *pos* if *n* == 0.

This function will insert a specified number of copies of the given data after the location specified by *pos*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 249 of file forward\_list.tcc.

**5.467.3.24** `template<typename _Tp, typename _Alloc > template<typename  
_InputIterator > forward_list< _Tp, _Alloc >::iterator  
forward_list::insert_after ( const_iterator __pos, _InputIterator  
__first, _InputIterator __last )`

Inserts a range into the forward\_list.

#### Parameters

*position* An iterator into the forward\_list.

*first* An input iterator.

*last* An input iterator.

#### Returns

An iterator pointing to the last inserted element or *pos* if *first* == *last*.

This function will insert copies of the data in the range [*first*,*last*) into the forward\_list after the location specified by *pos*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 264 of file forward\_list.tcc.

References std::forward\_list< \_Tp, \_Alloc >::empty().

**5.467.3.25** `template<typename _Tp, typename _Alloc > forward_list< _Tp, _Alloc >::iterator forward_list::insert_after ( const_iterator __pos, std::initializer_list< _Tp > __il )`

Inserts the contents of an [initializer\\_list](#) into forward\_list after the specified iterator.

#### Parameters

*pos* An iterator into the forward\_list.

*il* An [initializer\\_list](#) of value\_type.

#### Returns

An iterator pointing to the last inserted element or *pos* if *il* is empty.

This function will insert copies of the data in the [initializer\\_list](#) *il* into the forward\_list before the location specified by *pos*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 277 of file forward\_list.tcc.

**5.467.3.26** `template<typename _Tp, typename _Alloc = allocator<_Tp>> size_type std::forward_list< _Tp, _Alloc >::max_size ( ) const [inline]`

Returns the largest possible size of forward\_list.

Definition at line 743 of file forward\_list.h.

**5.467.3.27** `template<typename _Tp, typename _Alloc > template<typename _Comp > void forward_list::merge ( forward_list< _Tp, _Alloc > && __list, _Comp __comp )`

Merge sorted lists according to comparison function.

**Parameters**

*list* Sorted list to merge.

*comp* Comparison function defining sort order.

Assumes that both *list* and this list are sorted according to *comp*. Merges elements of *list* into this list in sorted order, leaving *list* empty when complete. Elements in this list precede elements in *list* that are equivalent according to *comp*().

Definition at line 357 of file forward\_list.tcc.

**5.467.3.28** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void  
std::forward_list< _Tp, _Alloc >::merge ( forward_list< _Tp,  
_Alloc > && __list ) [inline]`

Merge sorted lists.

**Parameters**

*list* Sorted list to merge.

Assumes that both *list* and this list are sorted according to *operator<()*. Merges elements of *list* into this list in sorted order, leaving *list* empty when complete. Elements in this list precede elements in *list* that are equal.

Definition at line 1155 of file forward\_list.h.

References std::forward\_list< \_Tp, \_Alloc >::merge().

Referenced by std::forward\_list< \_Tp, \_Alloc >::merge().

**5.467.3.29** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
forward_list& std::forward_list< _Tp, _Alloc >::operator= (  
forward_list< _Tp, _Alloc > && __list ) [inline]`

The forward\_list move assignment operator.

**Parameters**

*list* A forward\_list of identical element and allocator types.

The contents of *list* are moved into this forward\_list (without copying). *list* is a valid, but unspecified forward\_list

Definition at line 569 of file forward\_list.h.

References std::forward\_list< \_Tp, \_Alloc >::clear(), and std::forward\_list< \_Tp, \_Alloc >::swap().

**5.467.3.30** `template<typename _Tp, typename _Alloc > forward_list< _Tp, _Alloc > & forward_list::operator= ( const forward_list< _Tp, _Alloc > & __list )`

The forward\_list assignment operator.

#### Parameters

*list* A forward\_list of identical element and allocator types.

All the elements of *list* are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 145 of file forward\_list.tcc.

References std::forward\_list< \_Tp, \_Alloc >::before\_begin(), std::forward\_list< \_Tp, \_Alloc >::begin(), std::forward\_list< \_Tp, \_Alloc >::cbegin(), std::forward\_list< \_Tp, \_Alloc >::cend(), std::forward\_list< \_Tp, \_Alloc >::end(), std::forward\_list< \_Tp, \_Alloc >::erase\_after(), and std::forward\_list< \_Tp, \_Alloc >::insert\_after().

**5.467.3.31** `template<typename _Tp, typename _Alloc = allocator<_Tp>> forward_list& std::forward_list< _Tp, _Alloc >::operator= ( std::initializer_list< _Tp > __il ) [inline]`

The forward\_list initializer list assignment operator.

#### Parameters

*il* An [initializer\\_list](#) of value\_type.

Replace the contents of the forward\_list with copies of the elements in the [initializer\\_list](#) *il*. This is linear in *il.size()*.

Definition at line 587 of file forward\_list.h.

References std::forward\_list< \_Tp, \_Alloc >::assign().

**5.467.3.32** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list< _Tp, _Alloc >::pop_front ( ) [inline]`

Removes first element.

This is a typical stack operation. It shrinks the forward\_list by one. Due to the nature of a forward\_list this operation can be done in constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop_front()` is called.

Definition at line 823 of file forward\_list.h.

```
5.467.3.33 template<typename _Tp, typename _Alloc = allocator<_Tp>> void
 std::forward_list< _Tp, _Alloc >::push_front (const _Tp & __val
) [inline]
```

Add data to the front of the forward\_list.

#### Parameters

*val* Data to be added.

This is a typical stack operation. The function creates an element at the front of the forward\_list and assigns the given data to it. Due to the nature of a forward\_list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 800 of file forward\_list.h.

References `std::forward_list< _Tp, _Alloc >::cbefore_begin()`.

```
5.467.3.34 template<typename _Tp, typename _Alloc > void
 forward_list::remove (const _Tp & __val)
```

Remove all elements equal to value.

#### Parameters

*val* The value to remove.

Removes every element in the list equal to *value*. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 291 of file forward\_list.tcc.

**5.467.3.35** `template<typename _Tp , typename _Alloc > template<typename  
_Pred > void forward_list::remove_if ( _Pred __pred )`

Remove all elements satisfying a predicate.

#### Parameters

*pred* Unary predicate function or object.

Removes every element in the list for which the predicate returns true. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 320 of file forward\_list.tcc.

**5.467.3.36** `template<typename _Tp , typename _Alloc > void  
forward_list::resize ( size_type __sz )`

Resizes the forward\_list to the specified number of elements.

#### Parameters

*sz* Number of elements the forward\_list should contain.

This function will resize the forward\_list to the specified number of elements. If the number is smaller than the forward\_list's current size the forward\_list is truncated, otherwise the forward\_list is extended and the new elements are default constructed.

Definition at line 190 of file forward\_list.tcc.

References std::forward\_list< \_Tp, \_Alloc >::before\_begin(), std::forward\_list< \_Tp, \_Alloc >::end(), and std::forward\_list< \_Tp, \_Alloc >::erase\_after().

**5.467.3.37** `template<typename _Tp , typename _Alloc > void  
forward_list::resize ( size_type __sz, const value_type & __val )`

Resizes the forward\_list to the specified number of elements.

#### Parameters

*sz* Number of elements the forward\_list should contain.



*val* Data with which new elements should be populated.

This function will resize the forward\_list to the specified number of elements. If the number is smaller than the forward\_list's current size the forward\_list is truncated, otherwise the forward\_list is extended and new elements are populated with given data.

Definition at line 209 of file forward\_list.tcc.

References std::forward\_list< \_Tp, \_Alloc >::before\_begin(), std::forward\_list< \_Tp, \_Alloc >::end(), std::forward\_list< \_Tp, \_Alloc >::erase\_after(), and std::forward\_list< \_Tp, \_Alloc >::insert\_after().

**5.467.3.38** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void  
std::forward_list< _Tp, _Alloc >::reverse ( ) [inline]`

Reverse the elements in list.

Reverse the order of elements in the list in linear time.

Definition at line 1199 of file forward\_list.h.

**5.467.3.39** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void  
std::forward_list< _Tp, _Alloc >::sort ( ) [inline]`

Sort the elements of the list.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

Definition at line 1180 of file forward\_list.h.

References std::forward\_list< \_Tp, \_Alloc >::sort().

Referenced by std::forward\_list< \_Tp, \_Alloc >::sort().

**5.467.3.40** `template<typename _Tp, class _Alloc > template<typename  
_Comp > void forward_list::sort ( _Comp __comp )`

Sort the [forward\\_list](#) using a comparison function.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

Definition at line 403 of file forward\_list.tcc.

**5.467.3.41** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void  
std::forward_list< _Tp, _Alloc >::splice_after ( const_iterator  
__pos, forward_list< _Tp, _Alloc > && __list ) [inline]`

Insert contents of another forward\_list.

#### Parameters

*pos* Iterator referencing the element to insert after.

*list* Source list.

The elements of *list* are inserted in constant time after the element referenced by *pos*.  
*list* becomes an empty list.

Requires this != *x*.

Definition at line 1041 of file forward\_list.h.

Referenced by std::forward\_list< \_Tp, \_Alloc >::splice\_after().

**5.467.3.42** `template<typename _Tp , typename _Alloc > void  
forward_list::splice_after ( const_iterator __pos, forward_list<  
_Tp, _Alloc > && __list, const_iterator __before, const_iterator  
__last )`

Insert range from another forward\_list.

#### Parameters

*pos* Iterator referencing the element to insert after.

*list* Source list.

*before* Iterator referencing before the start of range in list.

*last* Iterator referencing the end of range in list.

Removes elements in the range (before,last) and inserts them after *pos* in constant time.

Undefined if *pos* is in (before,last).

Definition at line 238 of file forward\_list.tcc.

**5.467.3.43** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void  
std::forward_list< _Tp, _Alloc >::splice_after ( const_iterator  
__pos, forward_list< _Tp, _Alloc > && __list, const_iterator __i  
) [inline]`

Insert element from another forward\_list.

#### Parameters

*pos* Iterator referencing the element to insert after.

*list* Source list.

*i* Iterator referencing the element before the element to move.

Removes the element in list *list* referenced by *i* and inserts it into the current list after *pos*.

Definition at line 1058 of file forward\_list.h.

References std::forward\_list< \_Tp, \_Alloc >::splice\_after().

**5.467.3.44** `template<typename _Tp, typename _Alloc = allocator<_Tp>>  
void std::forward_list< _Tp, _Alloc >::swap ( forward_list< _Tp,  
_Alloc > & __list ) [inline]`

Swaps data with another forward\_list.

#### Parameters

*list* A forward\_list of the same element and allocator types.

This exchanges the elements between two lists in constant time. Note that the global std::swap() function is specialized such that std::swap(l1,l2) will feed to this function.

Definition at line 982 of file forward\_list.h.

Referenced by std::forward\_list< \_Tp, \_Alloc >::operator=(), and std::swap().

**5.467.3.45** `template<typename _Tp, typename _Alloc = allocator<_Tp>> void  
std::forward_list< _Tp, _Alloc >::unique ( ) [inline]`

Remove consecutive duplicate elements.

For each consecutive set of elements with the same value, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1126 of file `forward_list.h`.

References `std::forward_list<_Tp, _Alloc>::unique()`.

Referenced by `std::forward_list<_Tp, _Alloc>::unique()`.

### 5.467.3.46 `template<typename _Tp, typename _Alloc> template<typename _BinPred> void forward_list::unique ( _BinPred __binary_pred )`

Remove consecutive elements satisfying a predicate.

#### Parameters

*binary\_pred* Binary predicate function or object.

For each consecutive set of elements `[first,last)` that satisfy `predicate(first,i)` where `i` is an iterator in `[first,last)`, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 336 of file `forward_list.tcc`.

References `std::forward_list<_Tp, _Alloc>::begin()`, `std::forward_list<_Tp, _Alloc>::end()`, and `std::forward_list<_Tp, _Alloc>::erase_after()`.

The documentation for this class was generated from the following files:

- [forward\\_list.h](#)
- [forward\\_list.tcc](#)

## 5.468 `std::fpos<_StateT>` Class Template Reference

Class representing stream positions.

#### Public Member Functions

- [fpos](#) ([streamoff](#) \_\_off)
- [operator streamoff](#) () const
- [fpos operator+](#) ([streamoff](#) \_\_off) const

- `fpos & operator+= (streamoff __off)`
- `fpos operator- (streamoff __off) const`
- `streamoff operator- (const fpos &__other) const`
- `fpos & operator-= (streamoff __off)`
- `void state (_StateT __st)`
- `_StateT state () const`

### 5.468.1 Detailed Description

`template<typename _StateT> class std::fpos<_StateT>`

Class representing stream positions. The standard places no requirements upon the template parameter `StateT`. In this implementation `StateT` must be `DefaultConstructible`, `CopyConstructible` and `Assignable`. The standard only requires that `fpos` should contain a member of type `StateT`. In this implementation it also contains an offset stored as a signed integer.

#### Parameters

*StateT* Type passed to and returned from `state()`.

Definition at line 114 of file `postypes.h`.

### 5.468.2 Constructor & Destructor Documentation

**5.468.2.1** `template<typename _StateT> std::fpos<_StateT>::fpos (streamoff __off) [inline]`

Construct position from offset.

Definition at line 135 of file `postypes.h`.

### 5.468.3 Member Function Documentation

**5.468.3.1** `template<typename _StateT> std::fpos<_StateT>::operator streamoff ( ) const [inline]`

Convert to `streamoff`.

Definition at line 139 of file `postypes.h`.

**5.468.3.2** `template<typename _StateT> fpos std::fpos<_StateT>::operator+  
( streamoff __off ) const [inline]`

Add position and offset.

Definition at line 180 of file postypes.h.

**5.468.3.3** `template<typename _StateT> fpos& std::fpos<_StateT>::operator+= ( streamoff __off ) [inline]`

Add offset to this position.

Definition at line 156 of file postypes.h.

**5.468.3.4** `template<typename _StateT> streamoff std::fpos<_StateT>::operator- ( const fpos<_StateT> & __other ) const [inline]`

Subtract position to return offset.

Definition at line 207 of file postypes.h.

**5.468.3.5** `template<typename _StateT> fpos std::fpos<_StateT>::operator- ( streamoff __off ) const [inline]`

Subtract offset from position.

Definition at line 194 of file postypes.h.

**5.468.3.6** `template<typename _StateT> fpos& std::fpos<_StateT>::operator-= ( streamoff __off ) [inline]`

Subtract offset from this position.

Definition at line 167 of file postypes.h.

## 5.469 `std::front_insert_iterator< _Container >` Class Template Reference 2652

**5.468.3.7** `template<typename _StateT> _StateT std::fpos< _StateT >::state ( ) const [inline]`

Return the last set value of *st*.

Definition at line 148 of file `postypes.h`.

**5.468.3.8** `template<typename _StateT> void std::fpos< _StateT >::state ( _StateT __st ) [inline]`

Remember the value of *st*.

Definition at line 143 of file `postypes.h`.

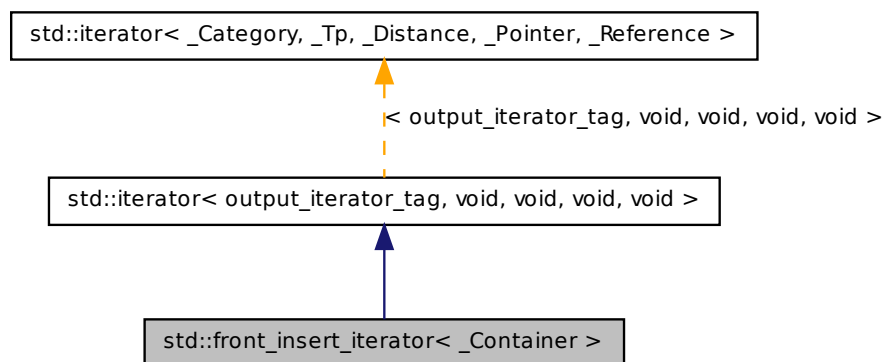
The documentation for this class was generated from the following file:

- [postypes.h](#)

## 5.469 `std::front_insert_iterator< _Container >` Class Template Reference

Turns assignment into insertion.

Inheritance diagram for `std::front_insert_iterator< _Container >`:



### Public Types

- typedef `_Container` `container_type`
- typedef void `difference_type`
- typedef `output_iterator_tag` `iterator_category`
- typedef void `pointer`
- typedef void `reference`
- typedef void `value_type`

### Public Member Functions

- `front_insert_iterator` (`_Container &__x`)
- `front_insert_iterator` & `operator*` ()
- `front_insert_iterator` & `operator++` ()
- `front_insert_iterator` `operator++` (int)
- `front_insert_iterator` & `operator=` (const typename `_Container::value_type` &\_\_value)
- `front_insert_iterator` & `operator=` (typename `_Container::value_type` &&\_\_value)

### Protected Attributes

- `_Container * container`

#### 5.469.1 Detailed Description

**template<typename `_Container`> class `std::front_insert_iterator<_Container>`**

Turns assignment into insertion. These are output iterators, constructed from a container-of-T. Assigning a T to the iterator prepends it to the container using `push_front`.

Tip: Using the `front_inserter` function to create these iterators can save typing.

Definition at line 487 of file `stl_iterator.h`.

#### 5.469.2 Member Typedef Documentation

**5.469.2.1 `template<typename _Container> typedef _Container std::front_insert_iterator<_Container>::container_type`**

A nested typedef for the type of whatever container you used.

Definition at line 495 of file `stl_iterator.h`.



**5.469.2.2** `typedef void std::iterator< output_iterator_tag , void , void , void ,  
void >::difference_type [inherited]`

Distance between iterators is represented as this type.

Definition at line 126 of file stl\_iterator\_base\_types.h.

**5.469.2.3** `typedef output_iterator_tag std::iterator< output_iterator_tag , void  
, void , void , void >::iterator_category [inherited]`

One of the [tag types](#).

Definition at line 122 of file stl\_iterator\_base\_types.h.

**5.469.2.4** `typedef void std::iterator< output_iterator_tag , void , void , void ,  
void >::pointer [inherited]`

This type represents a pointer-to-value\_type.

Definition at line 128 of file stl\_iterator\_base\_types.h.

**5.469.2.5** `typedef void std::iterator< output_iterator_tag , void , void , void ,  
void >::reference [inherited]`

This type represents a reference-to-value\_type.

Definition at line 130 of file stl\_iterator\_base\_types.h.

**5.469.2.6** `typedef void std::iterator< output_iterator_tag , void , void , void ,  
void >::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 124 of file stl\_iterator\_base\_types.h.

### **5.469.3 Constructor & Destructor Documentation**

**5.469.3.1** `template<typename _Container> std::front_insert_iterator<  
_Container>::front_insert_iterator ( _Container & __x )  
[inline, explicit]`

The only way to create this iterator is with a container.

Definition at line 498 of file `stl_iterator.h`.

### **5.469.4 Member Function Documentation**

**5.469.4.1** `template<typename _Container> front_insert_iterator&  
std::front_insert_iterator<_Container>::operator* ( )  
[inline]`

Simply returns `*this`.

Definition at line 536 of file `stl_iterator.h`.

**5.469.4.2** `template<typename _Container> front_insert_iterator  
std::front_insert_iterator<_Container>::operator++ ( int )  
[inline]`

Simply returns `*this`. (This iterator does not *move*.).

Definition at line 546 of file `stl_iterator.h`.

**5.469.4.3** `template<typename _Container> front_insert_iterator&  
std::front_insert_iterator<_Container>::operator++ ( )  
[inline]`

Simply returns `*this`. (This iterator does not *move*.).

Definition at line 541 of file `stl_iterator.h`.

**5.469.4.4** `template<typename _Container > front_insert_iterator&  
std::front_insert_iterator< _Container >::operator= ( const  
typename _Container::value_type & __value ) [inline]`

#### Parameters

*value* An instance of whatever type `container_type::const_reference` is; presumably a reference-to-const T for `container<T>`.

#### Returns

This iterator, for chained operations.

This kind of iterator doesn't really have a *position* in the container (you can think of the position as being permanently at the front, if you like). Assigning a value to the iterator will always prepend the value to the front of the container.

Definition at line 520 of file `stl_iterator.h`.

The documentation for this class was generated from the following file:

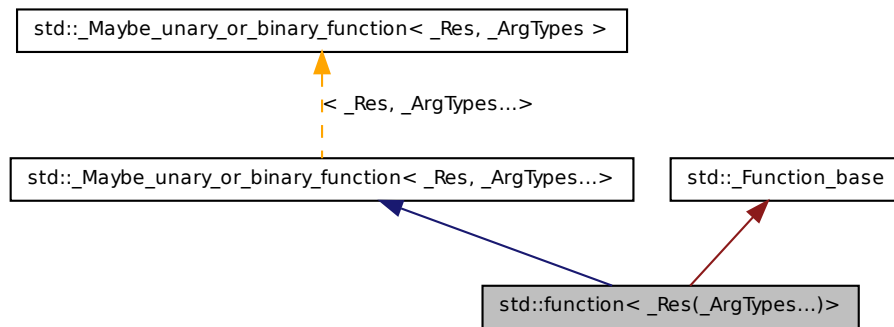
- [stl\\_iterator.h](#)

## **5.470 `std::function< _Res(_ArgTypes...)>` Class Template Reference**

Primary class template for `std::function`.

Polymorphic function wrapper.

Inheritance diagram for `std::function<_Res(_ArgTypes...)>`:



## Public Types

- typedef `_Res` **result\_type**

## Public Member Functions

- `function()`
- `function(nullptr_t)`
- `function(function &&__x)`
- `template<typename _Functor>`  
`function(_Functor __f, typename enable_if<!is_integral<_Functor>::value, _Useless>::type=_Useless())`
- `function(const function &&__x)`
- `operator bool() const`
- `_Res operator()(_ArgTypes... __args) const`
- `function & operator= (nullptr_t)`
- `template<typename _Functor>`  
`enable_if<!is_integral<_Functor>::value, function &>::type operator= (_Functor &&__f)`
- `template<typename _Functor>`  
`enable_if<!is_integral<_Functor>::value, function &>::type operator= (reference_wrapper<_Functor> __f)`
- `function & operator= (function &&__x)`
- `function & operator= (const function &&__x)`

- void `swap` (function &\_\_x)
- template<typename \_Functor >  
\_Functor \* `target` ()
- template<typename \_Functor >  
const \_Functor \* `target` () const
- const `type_info` & `target_type` () const

### Private Types

- typedef bool(\* `_Manager_type` )(\_Any\_data &, const \_Any\_data &, \_-  
Manager\_operation)

### Private Member Functions

- bool `_M_empty` () const

### Private Attributes

- \_Any\_data `_M_functor`
- \_Manager\_type `_M_manager`

### Static Private Attributes

- static const std::size\_t `_M_max_align`
- static const std::size\_t `_M_max_size`

#### 5.470.1 Detailed Description

`template<typename _Res, typename... _ArgTypes> class std::function< _Res(_-  
ArgTypes...)>`

Primary class template for `std::function`.

Polymorphic function wrapper.

Definition at line 1864 of file `functional`.

#### 5.470.2 Constructor & Destructor Documentation

**5.470.2.1** `template<typename _Res , typename... _ArgTypes> std::function<  
_Res(_ArgTypes...)>::function ( ) [inline]`

Default construct creates an empty function call wrapper.

**Postcondition**

`!(bool)*this`

Definition at line 1881 of file functional.

**5.470.2.2** `template<typename _Res , typename... _ArgTypes> std::function<  
_Res(_ArgTypes...)>::function ( nullptr_t ) [inline]`

Creates an empty function call wrapper.

**Postcondition**

`!(bool)*this`

Definition at line 1887 of file functional.

**5.470.2.3** `template<typename _Res , typename... _ArgTypes>  
std::function<_Res(_ArgTypes...)>::function ( const function<  
_Res(_ArgTypes...)> & __x )`

Function copy constructor.

**Parameters**

*x* A function object with identical call signature.

**Postcondition**

`(bool)*this == (bool)x`

The newly-created function contains a copy of the target of *x* (if it has one).

Definition at line 2112 of file functional.

**5.470.2.4** `template<typename _Res , typename... _ArgTypes> std::function<  
_Res(_ArgTypes...)>::function ( function<_Res(_ArgTypes...)> &&  
__x ) [inline]`

Function move constructor.

**Parameters**

*x* A function object rvalue with identical call signature.

The newly-created function contains the target of *x* (if it has one).

Definition at line 1906 of file functional.

```
5.470.2.5 template<typename _Res , typename... _ArgTypes>
 template<typename _Functor > std::function<
 _Res(_ArgTypes...)>::function (_Functor __f, typename enable_if<
 !is_integral< _Functor >::value, _Useless >::type = _Useless ())
```

Builds a function that targets a copy of the incoming function object.

**Parameters**

*f* A function object that is callable with parameters of type T1, T2, ..., TN and returns a value convertible to *Res*.

The newly-created function object will target a copy of *f*. If *f* is `reference_wrapper<F>`, then this function object will contain a reference to the function object `f.get()`. If *f* is a NULL function pointer or NULL pointer-to-member, the newly-created object will be empty.

If *f* is a non-NULL function pointer or an object of type `reference_wrapper<F>`, this function will not throw.

Definition at line 2126 of file functional.

**5.470.3 Member Function Documentation**

```
5.470.3.1 template<typename _Res , typename... _ArgTypes> std::function<
 _Res(_ArgTypes...)>::operator bool () const [inline,
 explicit]
```

Determine if the function wrapper has a target.

**Returns**

`true` when this function object contains a target, or `false` when it is empty.

This function will not throw an exception.

Definition at line 2061 of file functional.

**5.470.3.2** `template<typename _Res , typename... _ArgTypes> _Res  
std::function<_Res(_ArgTypes...)>::operator() ( _ArgTypes...  
__args ) const`

Invokes the function targeted by `*this`.

#### Returns

the result of the target.

#### Exceptions

*bad\_function\_call* when `!(bool)*this`

The function call operator invokes the target function object stored by `this`.

Definition at line 2144 of file `functional`.

**5.470.3.3** `template<typename _Res , typename... _ArgTypes> function&  
std::function<_Res(_ArgTypes...)>::operator= ( function<  
_Res(_ArgTypes...)> && __x ) [inline]`

Function move-assignment operator.

#### Parameters

`x` A function rvalue with identical call signature.

#### Returns

`*this`

The target of `x` is moved to `*this`. If `x` has no target, then `*this` will be empty.

If `x` targets a function pointer or a reference to a function object, then this operation will not throw an exception.

Definition at line 1966 of file `functional`.

**5.470.3.4** `template<typename _Res , typename... _ArgTypes>  
template<typename _Functor > enable_if<!is_integral<_  
_Functor>::value, function&>::type std::function<  
_Res(_ArgTypes...)>::operator= ( _Functor && __f ) [inline]`



Function assignment to a new target.

### Parameters

*f* A function object that is callable with parameters of type *T1*, *T2*, ..., *TN* and returns a value convertible to *Res*.

### Returns

`*this`

This function object wrapper will target a copy of *f*. If *f* is `reference_wrapper<F>`, then this function object will contain a reference to the function object `f.get()`. If *f* is a NULL function pointer or NULL pointer-to-member, `this` object will be empty.

If *f* is a non-NULL function pointer or an object of type `reference_wrapper<F>`, this function will not throw.

Definition at line 2009 of file `functional`.

```
5.470.3.5 template<typename _Res , typename... _ArgTypes>
 template<typename _Functor > enable_if<!is_integral<_
 Functor>::value, function&>::type std::function<
 _Res(_ArgTypes...)>::operator= (reference_wrapper< _Functor >
 __f) [inline]
```

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 2018 of file `functional`.

```
5.470.3.6 template<typename _Res , typename... _ArgTypes> function&
 std::function< _Res(_ArgTypes...)>::operator= (nullptr_t)
 [inline]
```

Function assignment to zero.

### Postcondition

`!(bool)*this`

### Returns

`*this`

The target of `*this` is deallocated, leaving it empty.

Definition at line 1980 of file `functional`.

**5.470.3.7** `template<typename _Res, typename... _ArgTypes> function&  
std::function<_Res(_ArgTypes...)>::operator= ( const function<  
_Res(_ArgTypes...)> & __x ) [inline]`

Function assignment operator.

#### Parameters

*x* A function with identical call signature.

#### Postcondition

`(bool)*this == (bool)x`

#### Returns

`*this`

The target of *x* is copied to `*this`. If *x* has no target, then `*this` will be empty.

If *x* targets a function pointer or a reference to a function object, then this operation will not throw an exception.

Definition at line 1948 of file `functional`.

**5.470.3.8** `template<typename _Res, typename... _ArgTypes> void  
std::function<_Res(_ArgTypes...)>::swap ( function<  
_Res(_ArgTypes...)> & __x ) [inline]`

Swap the targets of two function objects.

#### Parameters

*f* A function with identical call signature.

Swap the targets of `this` function object and *f*. This function will not throw an exception.

Definition at line 2033 of file `functional`.

**5.470.3.9** `template<typename _Res , typename... _ArgTypes>  
template<typename _Functor > _Functor * std::function<  
_Res(_ArgTypes...)>::target ( )`

Access the stored target function object.

#### Returns

Returns a pointer to the stored target function object, if `typeid(Functor).equals(target_type())`; otherwise, a NULL pointer.

This function will not throw an exception.

Definition at line 2171 of file functional.

**5.470.3.10** `template<typename _Res , typename... _ArgTypes>  
template<typename _Functor > const _Functor * std::function<  
_Res(_ArgTypes...)>::target ( ) const`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 2190 of file functional.

**5.470.3.11** `template<typename _Res , typename... _ArgTypes> const type_info  
& std::function<_Res(_ArgTypes...)>::target_type ( ) const`

Determine the type of the target of this function object wrapper.

#### Returns

the type identifier of the target function object, or `typeid(void)` if `!(bool)*this`.

This function will not throw an exception.

Definition at line 2155 of file functional.

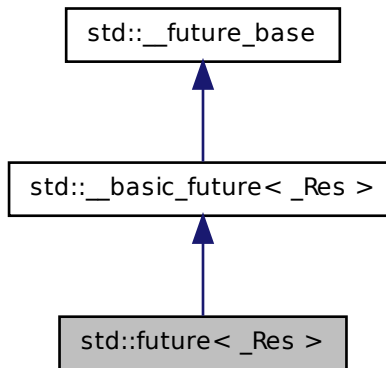
The documentation for this class was generated from the following file:

- [functional](#)

**5.471 `std::future< _Res >` Class Template Reference**

Primary template for future.

Inheritance diagram for `std::future< _Res >`:

**Public Member Functions**

- `future` (`future` &&\_\_uf)
- `future` (const `future` &)
- `_Res` `get` ()
- `future` & `operator=` (const `future` &)
- `future` & `operator=` (`future` &&\_\_fut)
- bool `valid` () const
- void `wait` () const
- template<typename `_Rep` , typename `_Period` >  
bool `wait_for` (const `chrono::duration`< `_Rep`, `_Period` > &\_\_rel) const
- template<typename `_Clock` , typename `_Duration` >  
bool `wait_until` (const `chrono::time_point`< `_Clock`, `_Duration` > &\_\_abs)  
const

**Static Public Member Functions**

- template<typename `_Res` , typename `_Allocator` >

```
static _Ptr< _Result_alloc< _Res, _Allocator > >::type _S_allocate_result
(const _Allocator &__a)
```

### Protected Types

- typedef [\\_\\_future\\_base::\\_Result](#)< \_Res > & [\\_\\_result\\_type](#)

### Protected Member Functions

- [\\_\\_result\\_type](#) [\\_M\\_get\\_result](#) ()
- void [\\_M\\_swap](#) ([\\_\\_basic\\_future](#) &\_\_that)

### Friends

- template<typename \_Fn, typename... \_Args>  
[future](#)< typename [result\\_of](#)< \_Fn(\_Args...)>::type > [async](#) ([launch](#), \_Fn &&, \_Args &&...)
- class [packaged\\_task](#)
- class [promise](#)< \_Res >

#### 5.471.1 Detailed Description

`template<typename _Res> class std::future< _Res >`

Primary template for future.

Definition at line 585 of file future.

#### 5.471.2 Constructor & Destructor Documentation

**5.471.2.1** `template<typename _Res > std::future< _Res >::future ( future< _Res > && __uf ) [inline]`

Move constructor.

Definition at line 603 of file future.

### 5.471.3 Member Function Documentation

#### 5.471.3.1 `template<typename _Res> __result_type std::__basic_future< _Res >::_M_get_result ( ) [inline, protected, inherited]`

Wait for the state to be ready and rethrow any stored exception.

Definition at line 538 of file `future`.

Referenced by `std::shared_future< _Res >::get()`, `std::future< void >::get()`, and `std::future< _Res >::get()`.

#### 5.471.3.2 `template<typename _Res > _Res std::future< _Res >::get ( ) [inline]`

Retrieving the value.

Definition at line 617 of file `future`.

References `std::__basic_future< _Res >::_M_get_result()`.

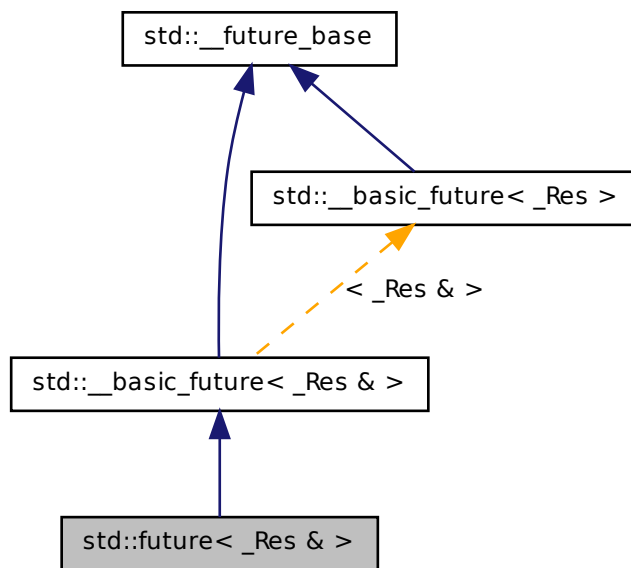
The documentation for this class was generated from the following file:

- [future](#)

## 5.472 `std::future< _Res & >` Class Template Reference

Partial specialization for `future<R&>`

Inheritance diagram for `std::future< _Res & >`:



### Public Member Functions

- `future` (`future` &&\_\_uf)
- `future` (const `future` &)
- `_Res` & `get` ()
- `future` & `operator=` (const `future` &)
- `future` & `operator=` (`future` &&\_\_fut)
- `bool valid` () const
- `void wait` () const
- `bool wait_for` (const `chrono::duration`< `_Rep`, `_Period` > &\_\_rel) const
- `bool wait_until` (const `chrono::time_point`< `_Clock`, `_Duration` > &\_\_abs) const

### Static Public Member Functions

- `template<typename _Res, typename _Allocator >`

```
static _Ptr< _Result_alloc< _Res, _Allocator > >::type _S_allocate_result
(const _Allocator &__a)
```

### Protected Types

- typedef `__future_base::_Result< _Res & > & __result_type`

### Protected Member Functions

- `__result_type _M_get_result ()`
- `void _M_swap ( __basic_future &__that)`

### Friends

- `template<typename _Fn, typename... _Args>`  
`future< typename result_of< _Fn(_Args...)>::type > async (launch, _Fn &&, _Args &&...)`
- class `packaged_task`
- class `promise< _Res & >`

#### 5.472.1 Detailed Description

`template<typename _Res> class std::future< _Res & >`

Partial specialization for `future<R&>`

Definition at line 626 of file `future`.

#### 5.472.2 Constructor & Destructor Documentation

**5.472.2.1** `template<typename _Res > std::future< _Res & >::future ( future< _Res & > && __uf ) [inline]`

Move constructor.

Definition at line 644 of file `future`.



### 5.472.3 Member Function Documentation

#### 5.472.3.1 `__result_type std::__basic_future< _Res & >::_M_get_result ( )` `[inline, protected, inherited]`

Wait for the state to be ready and rethrow any stored exception.

Definition at line 538 of file `future`.

References `std::rethrow_exception()`.

#### 5.472.3.2 `template<typename _Res > _Res& std::future< _Res & >::get ( )` `[inline]`

Retrieving the value.

Definition at line 658 of file `future`.

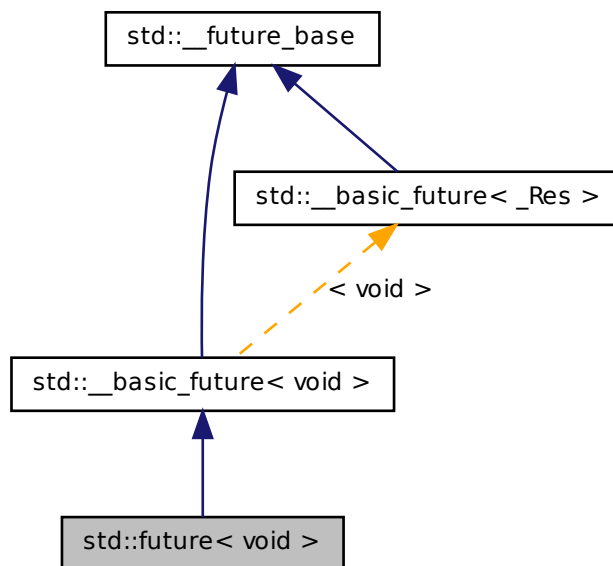
The documentation for this class was generated from the following file:

- [future](#)

## 5.473 `std::future< void >` Class Template Reference

Explicit specialization for [future<void>](#)

Inheritance diagram for `std::future< void >`:



### Public Member Functions

- `future` (`future` &&\_\_uf)
- `future` (const `future` &)
- void `get` ()
- `future` & `operator=` (const `future` &)
- `future` & `operator=` (`future` &&\_\_fut)
- bool `valid` () const
- void `wait` () const
- bool `wait_for` (const `chrono::duration`<\_Rep, \_Period> &\_\_rel) const
- bool `wait_until` (const `chrono::time_point`<\_Clock, \_Duration> &\_\_abs) const

### Static Public Member Functions

- `template<typename _Res, typename _Allocator>`

```
static _Ptr< _Result_alloc< _Res, _Allocator > >::type _S_allocate_result
(const _Allocator &__a)
```

### Protected Types

- typedef `__future_base::_Result< void > & __result_type`

### Protected Member Functions

- `__result_type _M_get_result ()`
- `void _M_swap ( __basic_future &__that)`

### Friends

- `template<typename _Fn, typename... _Args>`  
`future< typename result_of< _Fn(_Args...)>::type > async (launch, _Fn &&, _Args &&...)`
- class `packaged_task`
- class `promise< void >`

#### 5.473.1 Detailed Description

`template<> class std::future< void >`

Explicit specialization for `future<void>`

Definition at line 667 of file future.

#### 5.473.2 Constructor & Destructor Documentation

**5.473.2.1** `std::future< void >::future ( future< void > && __uf )`  
`[inline]`

Move constructor.

Definition at line 685 of file future.

### 5.473.3 Member Function Documentation

#### 5.473.3.1 `__result_type std::__basic_future< void >::_M_get_result ( )` `[inline, protected, inherited]`

Wait for the state to be ready and rethrow any stored exception.

Definition at line 538 of file `future`.

#### 5.473.3.2 `void std::future< void >::get ( ) [inline]`

Retrieving the value.

Definition at line 699 of file `future`.

References `std::__basic_future< _Res >::_M_get_result()`.

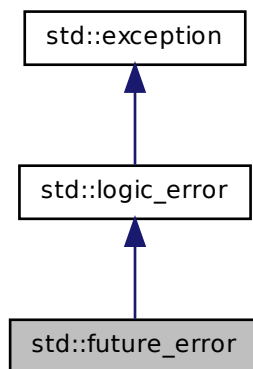
The documentation for this class was generated from the following file:

- [future](#)

## 5.474 `std::future_error` Class Reference

Exception type thrown by futures.

Inheritance diagram for `std::future_error`:



### Public Member Functions

- **future\_error** ([error\\_code](#) \_\_ec)
- const [error\\_code](#) & **code** () const throw ()
- virtual const char \* **what** () const throw ()

#### 5.474.1 Detailed Description

Exception type thrown by futures.

Definition at line 91 of file `future`.

#### 5.474.2 Member Function Documentation

##### 5.474.2.1 virtual const char\* std::future\_error::what ( ) const throw () [virtual]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::logic\\_error](#).

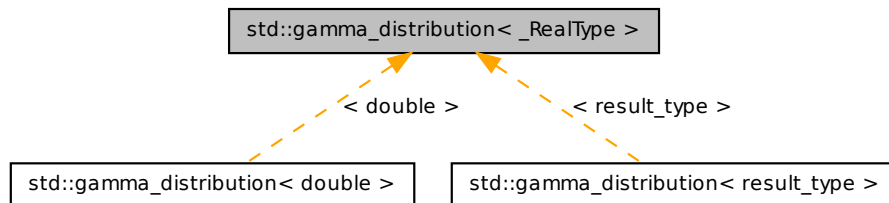
The documentation for this class was generated from the following file:

- [future](#)

## 5.475 `std::gamma_distribution<_RealType>` Class Template Reference

A gamma continuous distribution for random numbers.

Inheritance diagram for `std::gamma_distribution<_RealType>`:



### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_RealType` [result\\_type](#)

### Public Member Functions

- [gamma\\_distribution](#) (`_RealType __alpha_val=_RealType(1), _RealType __beta_val=_RealType(1)`)
- [gamma\\_distribution](#) (const [param\\_type](#) &\_\_p)
- `_RealType alpha () const`
- `_RealType beta () const`
- [result\\_type max \(\) const](#)
- [result\\_type min \(\) const](#)
- `template<typename _UniformRandomNumberGenerator>`  
[result\\_type operator\(\)](#) (`_UniformRandomNumberGenerator &__urng, const param\_type &__p`)

- `template<typename _UniformRandomNumberGenerator >`  
`result_type operator() (_UniformRandomNumberGenerator &__urng)`
- `void param (const param_type &__param)`
- `param_type param () const`
- `void reset ()`

## Friends

- `template<typename _RealType1, typename _CharT, typename _Traits >`  
`std::basic_ostream<_CharT, _Traits> & operator<< (std::basic_ostream<_CharT, _Traits> &, const std::gamma_distribution<_RealType1> &)`
- `template<typename _RealType1 >`  
`bool operator== (const std::gamma_distribution<_RealType1> &__d1, const std::gamma_distribution<_RealType1> &__d2)`
- `template<typename _RealType1, typename _CharT, typename _Traits >`  
`std::basic_istream<_CharT, _Traits> & operator>> (std::basic_istream<_CharT, _Traits> &, std::gamma_distribution<_RealType1> &)`

### 5.475.1 Detailed Description

`template<typename _RealType = double> class std::gamma_distribution<_RealType>`

A gamma continuous distribution for random numbers. The formula for the gamma probability density function is:

$$p(x|\alpha, \beta) = \frac{1}{\beta\Gamma(\alpha)} (x/\beta)^{\alpha-1} e^{-x/\beta}$$

Definition at line 2327 of file random.h.

### 5.475.2 Member Typedef Documentation

**5.475.2.1** `template<typename _RealType = double> typedef _RealType std::gamma_distribution<_RealType>::result_type`

The type of the range of the distribution.

Definition at line 2334 of file random.h.

### 5.475.3 Constructor & Destructor Documentation

**5.475.3.1** `template<typename _RealType = double> std::gamma_distribution<_RealType>::gamma_distribution ( _RealType __alpha_val  
= _RealType(1), _RealType __beta_val = _RealType(1) )  
[inline, explicit]`

Constructs a gamma distribution with parameters  $\alpha$  and  $\beta$ .

Definition at line 2379 of file random.h.

### 5.475.4 Member Function Documentation

**5.475.4.1** `template<typename _RealType = double> _RealType  
std::gamma_distribution<_RealType>::alpha ( ) const  
[inline]`

Returns the  $\alpha$  of the distribution.

Definition at line 2400 of file random.h.

**5.475.4.2** `template<typename _RealType = double> _RealType  
std::gamma_distribution<_RealType>::beta ( ) const [inline]`

Returns the  $\beta$  of the distribution.

Definition at line 2407 of file random.h.

**5.475.4.3** `template<typename _RealType = double> result_type  
std::gamma_distribution<_RealType>::max ( ) const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 2436 of file random.h.

Referenced by `std::gamma_distribution<result_type>::max()`.



**5.475.4.4** `template<typename _RealType = double> result_type  
std::gamma_distribution<_RealType>::min ( ) const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 2429 of file random.h.

**5.475.4.5** `template<typename _RealType = double> template<typename  
_UniformRandomNumberGenerator > result_type  
std::gamma_distribution<_RealType>::operator() (   
_UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 2444 of file random.h.

Referenced by `std::gamma_distribution<result_type>::operator()()`.

**5.475.4.6** `template<typename _RealType > template<typename  
_UniformRandomNumberGenerator > gamma_distribution<  
_RealType>::result_type std::gamma_distribution<_RealType  
>::operator() ( _UniformRandomNumberGenerator & __urng,  
const param_type & __param )`

Marsaglia, G. and Tsang, W. W. "A Simple Method for Generating Gamma Variables" ACM Transactions on Mathematical Software, 26, 3, 363-372, 2000.

Definition at line 2017 of file random.tcc.

References `std::log()`, and `std::pow()`.

**5.475.4.7** `template<typename _RealType = double> void  
std::gamma_distribution<_RealType>::param ( const param_type  
& __param ) [inline]`

Sets the parameter set of the distribution.

#### Parameters

**`__param`** The new parameter set of the distribution.

Definition at line 2422 of file random.h.

**5.475.4.8** `template<typename _RealType = double> param_type  
std::gamma_distribution<_RealType>::param ( ) const  
[inline]`

Returns the parameter set of the distribution.

Definition at line 2414 of file random.h.

Referenced by `std::gamma_distribution<result_type>::operator()()`.

**5.475.4.9** `template<typename _RealType = double> void  
std::gamma_distribution<_RealType>::reset ( ) [inline]`

Resets the distribution state.

Definition at line 2393 of file random.h.

Referenced by `std::negative_binomial_distribution<_IntType>::reset()`, `std::student_t_distribution<_RealType>::reset()`, `std::fisher_f_distribution<_RealType>::reset()`, and `std::chi_squared_distribution<_RealType>::reset()`.

## 5.475.5 Friends And Related Function Documentation

**5.475.5.1** `template<typename _RealType = double> template<typename  
_RealType1 , typename _CharT , typename _Traits >  
std::basic_ostream<_CharT, _Traits>& operator<<  
( std::basic_ostream<_CharT, _Traits> &, const  
std::gamma_distribution<_RealType1> & ) [friend]`

Inserts a `gamma_distribution` random number distribution `__x` into the output stream `__os`.

### Parameters

`__os` An output stream.

`__x` A `gamma_distribution` random number distribution.

### Returns

The output stream with the state of `__x` inserted or in an error state.

## 5.476 `std::gamma_distribution<_RealType>::param_type` Struct Reference 2680

**5.475.5.2** `template<typename _RealType = double> template<typename  
_RealType1 > bool operator==( const std::gamma_distribution<  
_RealType1 > & __d1, const std::gamma_distribution< _RealType1  
> & __d2 ) [friend]`

Return true if two gamma distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 2459 of file random.h.

**5.475.5.3** `template<typename _RealType = double> template<typename  
_RealType1 , typename _CharT , typename _Traits >  
std::basic_istream<_CharT, _Traits>& operator>>  
( std::basic_istream< _CharT, _Traits > & ,  
std::gamma_distribution< _RealType1 > & ) [friend]`

Extracts a gamma\_distribution random number distribution `__x` from the input stream `__is`.

### Parameters

`__is` An input stream.

`__x` A gamma\_distribution random number generator engine.

### Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.476 `std::gamma_distribution< _RealType >::param_type` Struct Reference

### Public Types

- typedef [gamma\\_distribution<\\_RealType>](#) `distribution_type`

## 5.477 `std::geometric_distribution< _IntType >` Class Template Reference 2681

---

### Public Member Functions

- `param_type` (`_RealType __alpha_val=_RealType(1), _RealType __beta_val=_RealType(1)`)
- `_RealType alpha` () const
- `_RealType beta` () const

### Friends

- class `gamma_distribution< _RealType >`
- bool `operator==` (const `param_type` &\_\_p1, const `param_type` &\_\_p2)

### 5.476.1 Detailed Description

`template<typename _RealType = double> struct std::gamma_distribution< _RealType >::param_type`

Parameter type.

Definition at line 2336 of file `random.h`.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.477 `std::geometric_distribution< _IntType >` Class Template Reference

A discrete geometric random number distribution.

### Classes

- struct `param_type`

### Public Types

- typedef `_IntType` `result_type`

**Public Member Functions**

- **geometric\_distribution** (double \_\_p=0.5)
- **geometric\_distribution** (const [param\\_type](#) &\_\_p)
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- template<typename \_UniformRandomNumberGenerator >  
  [result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator >  
  [result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- double **p** () const
- void **param** (const [param\\_type](#) &\_\_param)
- [param\\_type](#) **param** () const
- void **reset** ()

**5.477.1 Detailed Description**

**template<typename \_IntType = int> class std::geometric\_distribution< \_IntType >**

A discrete geometric random number distribution. The formula for the geometric probability density function is  $p(i|p) = (1 - p)p^{i-1}$  where  $p$  is the parameter of the distribution.

Definition at line 3596 of file random.h.

**5.477.2 Member Typedef Documentation**

**5.477.2.1** **template<typename \_IntType = int> typedef \_IntType  
std::geometric\_distribution< \_IntType >::result\_type**

The type of the range of the distribution.

Definition at line 3603 of file random.h.

**5.477.3 Member Function Documentation**

**5.477.3.1** **template<typename \_IntType = int> result\_type  
std::geometric\_distribution< \_IntType >::max ( ) const  
[inline]**

Returns the least upper bound value of the distribution.

## 5.477 `std::geometric_distribution<_IntType>` Class Template Reference 2683

---

Definition at line 3689 of file random.h.

**5.477.3.2** `template<typename _IntType = int> result_type  
std::geometric_distribution< _IntType >::min ( ) const  
[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 3682 of file random.h.

**5.477.3.3** `template<typename _IntType = int> template<typename  
_UniformRandomNumberGenerator > result_type  
std::geometric_distribution< _IntType >::operator() (   
_UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 3697 of file random.h.

References `std::geometric_distribution< _IntType >::operator()()`, and  
`std::geometric_distribution< _IntType >::param()`.

Referenced by `std::geometric_distribution< _IntType >::operator()()`.

**5.477.3.4** `template<typename _IntType = int> double  
std::geometric_distribution< _IntType >::p ( ) const [inline]`

Returns the distribution parameter `p`.

Definition at line 3660 of file random.h.

**5.477.3.5** `template<typename _IntType = int> void std::geometric_  
distribution< _IntType >::param ( const param_type & __param )  
[inline]`

Sets the parameter set of the distribution.

### Parameters

`__param` The new parameter set of the distribution.

## 5.478 `std::geometric_distribution< _IntType >::param_type` Struct Reference 2684

---

Definition at line 3675 of file `random.h`.

**5.477.3.6** `template<typename _IntType = int> param_type  
std::geometric_distribution< _IntType >::param ( ) const  
[inline]`

Returns the parameter set of the distribution.

Definition at line 3667 of file `random.h`.

Referenced by `std::geometric_distribution< _IntType >::operator()()`, `std::operator==( )`, and `std::operator>>()`.

**5.477.3.7** `template<typename _IntType = int> void  
std::geometric_distribution< _IntType >::reset ( ) [inline]`

Resets the distribution state.

Does nothing for the geometric distribution.

Definition at line 3654 of file `random.h`.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.478 `std::geometric_distribution< _IntType >::param_type` Struct Reference

### Public Types

- typedef [geometric\\_distribution< \\_IntType >](#) `distribution_type`

### Public Member Functions

- `param_type` (double \_\_p=0.5)
- `double p () const`

### Friends

- class `geometric_distribution< _IntType >`

- `bool operator==(const param\_type &__p1, const param\_type &__p2)`

#### 5.478.1 Detailed Description

`template<typename _IntType = int> struct std::geometric_distribution< _IntType >::param_type`

Parameter type.

Definition at line 3605 of file `random.h`.

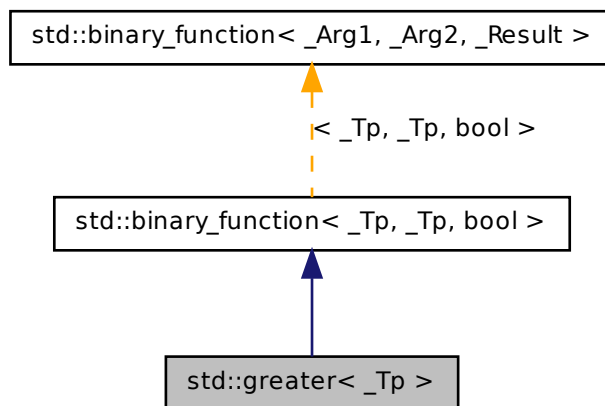
The documentation for this struct was generated from the following file:

- [random.h](#)

### 5.479 std::greater< \_Tp > Struct Template Reference

One of the [comparison functors](#).

Inheritance diagram for `std::greater< _Tp >`:



#### Public Types

- `typedef _Tp first\_argument\_type`



- typedef bool [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

### Public Member Functions

- bool **operator()** (const `_Tp` &\_\_x, const `_Tp` &\_\_y) const

#### 5.479.1 Detailed Description

`template<typename _Tp> struct std::greater<_Tp>`

One of the [comparison functors](#).

Definition at line 223 of file `stl_function.h`.

#### 5.479.2 Member Typedef Documentation

**5.479.2.1** `typedef _Tp std::binary_function<_Tp, _Tp, bool  
>::first_argument_type [inherited]`

`first_argument_type` is the type of the first argument

Definition at line 118 of file `stl_function.h`.

**5.479.2.2** `typedef bool std::binary_function<_Tp, _Tp, bool >::result_type  
[inherited]`

`result_type` is the return type

Definition at line 124 of file `stl_function.h`.

**5.479.2.3** `typedef _Tp std::binary_function<_Tp, _Tp, bool  
>::second_argument_type [inherited]`

`second_argument_type` is the type of the second argument

Definition at line 121 of file `stl_function.h`.

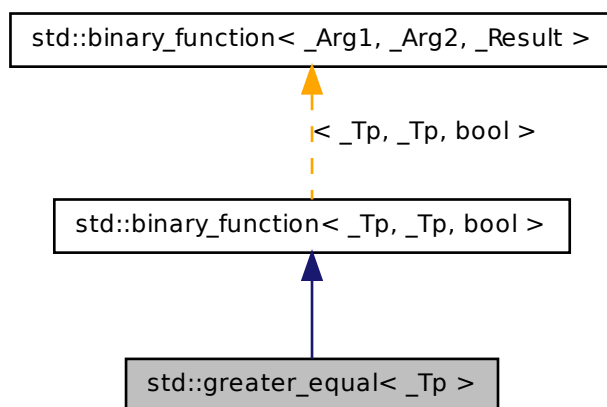
The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.480 std::greater\_equal< \_Tp > Struct Template Reference

One of the [comparison functions](#).

Inheritance diagram for std::greater\_equal< \_Tp >:



### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

### Public Member Functions

- `bool operator() (const _Tp &__x, const _Tp &__y) const`

#### 5.480.1 Detailed Description

**template<typename \_Tp> struct std::greater\_equal< \_Tp >**

One of the [comparison functions](#).

Definition at line 241 of file `stl_function.h`.

### 5.480.2 Member Typedef Documentation

**5.480.2.1** `typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type [inherited]`

`first_argument_type` is the type of the first argument

Definition at line 118 of file `stl_function.h`.

**5.480.2.2** `typedef bool std::binary_function< _Tp , _Tp , bool >::result_type [inherited]`

`result_type` is the return type

Definition at line 124 of file `stl_function.h`.

**5.480.2.3** `typedef _Tp std::binary_function< _Tp , _Tp , bool >::second_argument_type [inherited]`

`second_argument_type` is the type of the second argument

Definition at line 121 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.481 `std::gslice` Class Reference

Class defining multi-dimensional subset of an array.

### Public Member Functions

- [gslice](#) ()
- [gslice](#) (size\_t, const [valarray](#)< size\_t > &, const [valarray](#)< size\_t > &)
- [gslice](#) (const [gslice](#) &)
- [~gslice](#) ()
- [gslice](#) & [operator=](#) (const [gslice](#) &)
- [valarray](#)< size\_t > [size](#) () const
- size\_t [start](#) () const
- [valarray](#)< size\_t > [stride](#) () const

## Friends

- class `valarray`

### 5.481.1 Detailed Description

Class defining multi-dimensional subset of an array. The slice class represents a multi-dimensional subset of an array, specified by three parameter sets: start offset, size array, and stride array. The start offset is the index of the first element of the array that is part of the subset. The size and stride array describe each dimension of the slice. Size is the number of elements in that dimension, and stride is the distance in the array between successive elements in that dimension. Each dimension's size and stride is taken to begin at an array element described by the previous dimension. The size array and stride array must be the same size.

For example, if you have `offset==3`, `stride[0]==11`, `size[1]==3`, `stride[1]==3`, then `slice[0,0]==array[3]`, `slice[0,1]==array[6]`, `slice[0,2]==array[9]`, `slice[1,0]==array[14]`, `slice[1,1]==array[17]`, `slice[1,2]==array[20]`.

Definition at line 65 of file `gslice.h`.

The documentation for this class was generated from the following file:

- [gslice.h](#)

## 5.482 `std::gslice_array<_Tp>` Class Template Reference

Reference to multi-dimensional subset of an array.

### Public Types

- typedef `_Tp value_type`

### Public Member Functions

- [gslice\\_array](#) (const [gslice\\_array](#) &)
- template<class `_Dom`>  
void **operator%=>** (const `_Expr<_Dom, _Tp>` &) const
- void **operator%=>** (const [valarray](#)<`_Tp`> &) const
- void **operator&=>** (const [valarray](#)<`_Tp`> &) const
- template<class `_Dom`>  
void **operator&=>** (const `_Expr<_Dom, _Tp>` &) const
- template<class `_Dom`>  
void **operator\*=>** (const `_Expr<_Dom, _Tp>` &) const

- void `operator*=(const valarray<_Tp> &)` const
- template<class \_Dom >  
void `operator+=(const _Expr<_Dom, _Tp> &)` const
- void `operator+=(const valarray<_Tp> &)` const
- void `operator-=(const valarray<_Tp> &)` const
- template<class \_Dom >  
void `operator-=(const _Expr<_Dom, _Tp> &)` const
- template<class \_Dom >  
void `operator/=(const _Expr<_Dom, _Tp> &)` const
- void `operator/=(const valarray<_Tp> &)` const
- void `operator<<=(const valarray<_Tp> &)` const
- template<class \_Dom >  
void `operator<<=(const _Expr<_Dom, _Tp> &)` const
- void `operator= (const _Tp &)` const
- `gslice_array & operator= (const gslice_array &)`
- template<class \_Dom >  
void `operator= (const _Expr<_Dom, _Tp> &)` const
- void `operator= (const valarray<_Tp> &)` const
- template<class \_Dom >  
void `operator>>= (const _Expr<_Dom, _Tp> &)` const
- void `operator>>= (const valarray<_Tp> &)` const
- template<class \_Dom >  
void `operator^= (const _Expr<_Dom, _Tp> &)` const
- void `operator^= (const valarray<_Tp> &)` const
- void `operator|= (const valarray<_Tp> &)` const
- template<class \_Dom >  
void `operator|= (const _Expr<_Dom, _Tp> &)` const

## Friends

- class `valarray<_Tp>`

### 5.482.1 Detailed Description

`template<typename _Tp> class std::gslice_array<_Tp>`

Reference to multi-dimensional subset of an array. A `gslice_array` is a reference to the actual elements of an array specified by a `gslice`. The way to get a `gslice_array` is to call `operator[] (gslice)` on a `valarray`. The returned `gslice_array` then permits carrying operations out on the referenced subset of elements in the original `valarray`. For example, `operator+=(valarray)` will add values to the subset of elements in the underlying `valarray` this `gslice_array` refers to.

### Parameters

*Tp* Element type.

Definition at line 61 of file `gslice_array.h`.

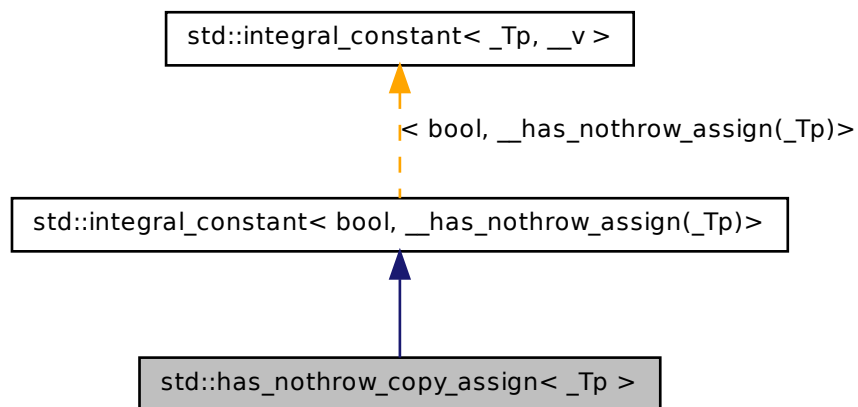
The documentation for this class was generated from the following file:

- [gslice\\_array.h](#)

### 5.483 `std::has_nothrow_copy_assign< _Tp >` Struct Template Reference

[has\\_nothrow\\_copy\\_assign](#)

Inheritance diagram for `std::has_nothrow_copy_assign< _Tp >`:



### Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

## 5.484 `std::has_nothrow_copy_constructor< _Tp >` Struct Template Reference 2692

---

### Public Member Functions

- `constexpr operator value_type ()`

### Static Public Attributes

- `static constexpr bool value`

#### 5.483.1 Detailed Description

`template<typename _Tp> struct std::has_nothrow_copy_assign< _Tp >`

[has\\_nothrow\\_copy\\_assign](#)

Definition at line 749 of file `type_traits`.

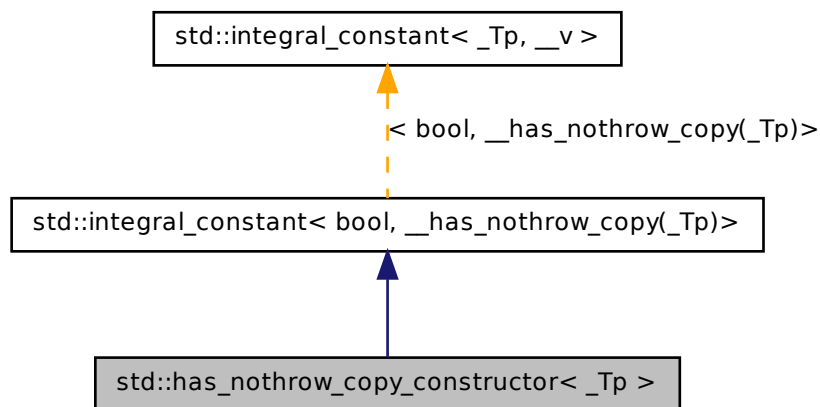
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.484 `std::has_nothrow_copy_constructor< _Tp >` Struct Template Reference

[has\\_nothrow\\_copy\\_constructor](#)

Inheritance diagram for `std::has_nothrow_copy_constructor< _Tp >`:



### Public Types

- typedef `integral_constant< bool, __v >` **type**
- typedef `bool` **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr `bool` **value**

#### 5.484.1 Detailed Description

`template<typename _Tp> struct std::has_nothrow_copy_constructor< _Tp >`

[`has\_nothrow\_copy\_constructor`](#)

Definition at line 743 of file `type_traits`.



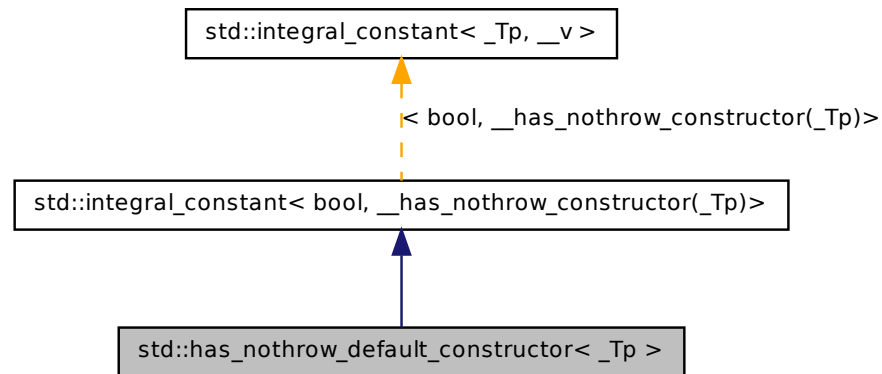
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.485 `std::has_nothrow_default_constructor< _Tp >` Struct Template Reference

[has\\_nothrow\\_default\\_constructor](#)

Inheritance diagram for `std::has_nothrow_default_constructor< _Tp >`:



### Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr bool **value**

## 5.485.1 Detailed Description

`template<typename _Tp> struct std::has_nothrow_default_constructor< _Tp >`

[has\\_nothrow\\_default\\_constructor](#)

Definition at line 737 of file `type_traits`.

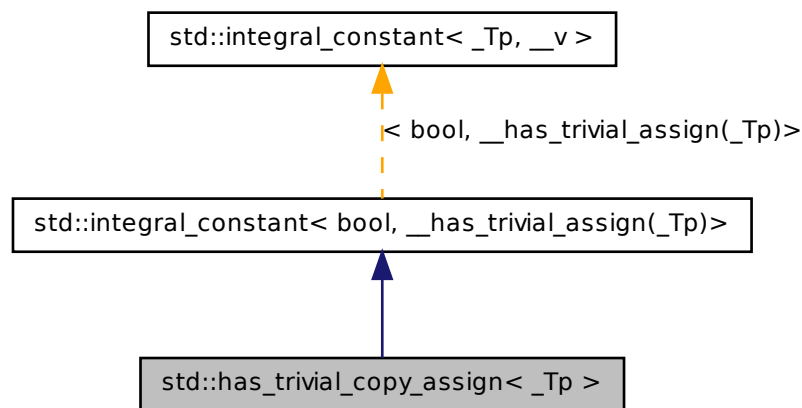
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

5.486 `std::has_trivial_copy_assign< _Tp >` Struct Template Reference

[has\\_trivial\\_copy\\_assign](#)

Inheritance diagram for `std::has_trivial_copy_assign< _Tp >`:



## Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

## 5.487 `std::has_trivial_copy_constructor< _Tp >` Struct Template Reference 2696

### Public Member Functions

- `constexpr operator value_type ()`

### Static Public Attributes

- `static constexpr bool value`

#### 5.486.1 Detailed Description

`template<typename _Tp> struct std::has_trivial_copy_assign< _Tp >`

[has\\_trivial\\_copy\\_assign](#)

Definition at line 725 of file `type_traits`.

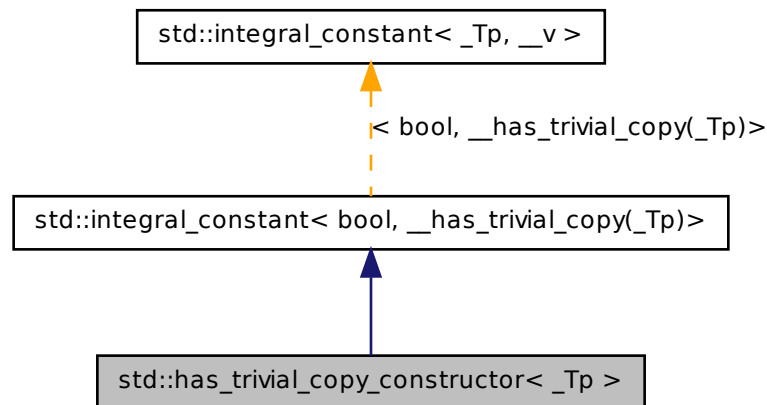
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.487 `std::has_trivial_copy_constructor< _Tp >` Struct Template Reference

[has\\_trivial\\_copy\\_constructor](#)

Inheritance diagram for `std::has_trivial_copy_constructor< _Tp >`:



### Public Types

- typedef `integral_constant< bool, __v >` **type**
- typedef `bool` **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr `bool` **value**

#### 5.487.1 Detailed Description

`template<typename _Tp> struct std::has_trivial_copy_constructor< _Tp >`

[`has\_trivial\_copy\_constructor`](#)

Definition at line 719 of file `type_traits`.

## 5.488 `std::has_trivial_default_constructor< _Tp >` Struct Template Reference

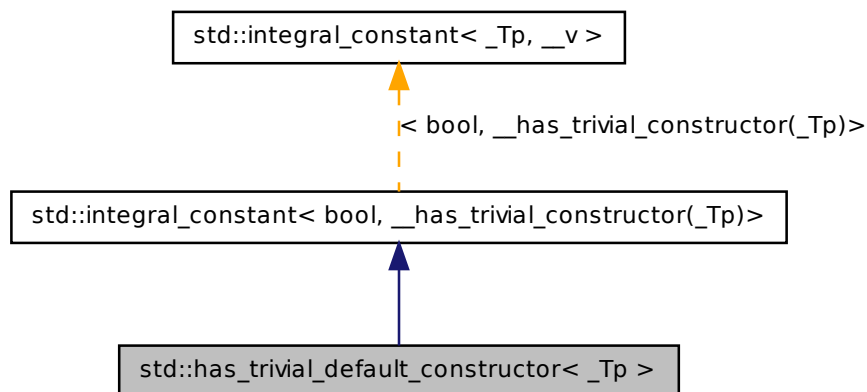
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.488 `std::has_trivial_default_constructor< _Tp >` Struct Template Reference

[has\\_trivial\\_default\\_constructor](#)

Inheritance diagram for `std::has_trivial_default_constructor< _Tp >`:



### Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr bool **value**

## 5.488.1 Detailed Description

`template<typename _Tp> struct std::has_trivial_default_constructor< _Tp >`

[has\\_trivial\\_default\\_constructor](#)

Definition at line 713 of file `type_traits`.

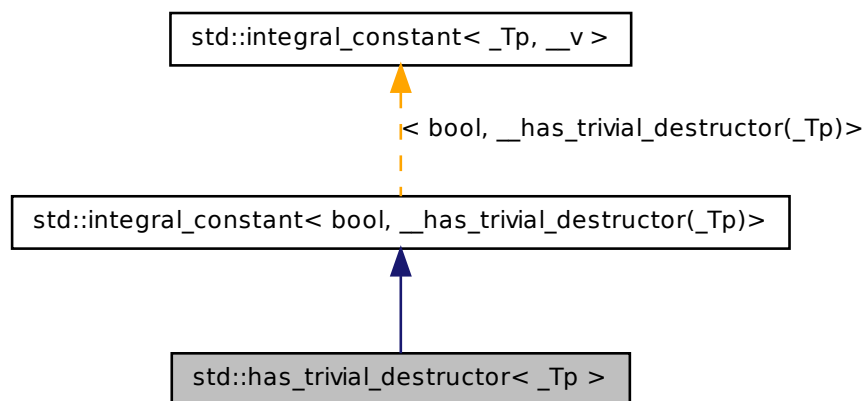
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

5.489 `std::has_trivial_destructor< _Tp >` Struct Template Reference

[has\\_trivial\\_destructor](#)

Inheritance diagram for `std::has_trivial_destructor< _Tp >`:



## Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

#### Public Member Functions

- `constexpr operator value_type ()`

#### Static Public Attributes

- `static constexpr bool value`

#### 5.489.1 Detailed Description

`template<typename _Tp> struct std::has_trivial_destructor<_Tp>`

[has\\_trivial\\_destructor](#)

Definition at line 731 of file `type_traits`.

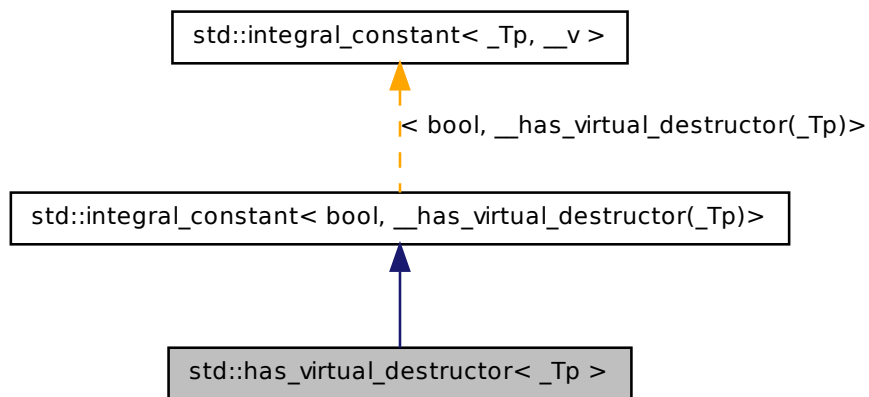
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### **5.490** `std::has_virtual_destructor<_Tp>` Struct Template Reference

[has\\_virtual\\_destructor](#)

Inheritance diagram for `std::has_virtual_destructor<_Tp>`:



### Public Types

- typedef `integral_constant< bool, __v>` **type**
- typedef `bool` **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr `bool` **value**

#### 5.490.1 Detailed Description

**template<typename \_Tp> struct `std::has_virtual_destructor<_Tp>`**

[`has\_virtual\_destructor`](#)

Definition at line 361 of file `type_traits`.

The documentation for this struct was generated from the following file:



- [type\\_traits](#)

## 5.491 `std::hash< _Tp >` Struct Template Reference

Primary class template hash.

Inherits `std::__hash_base< size_t, _Tp >`.

### Public Types

- `typedef _Arg` **argument\_type**
- `typedef _Result` **result\_type**

### Public Member Functions

- `size_t operator() (_Tp __val) const`
- `template<>`  
`size_t operator() (long double __val) const`
- `template<>`  
`size_t operator() (double __val) const`
- `template<>`  
`size_t operator() (float __val) const`
- `template<>`  
`size_t operator() (unsigned long long __val) const`
- `template<>`  
`size_t operator() (unsigned long __val) const`
- `template<>`  
`size_t operator() (unsigned int __val) const`
- `template<>`  
`size_t operator() (unsigned short __val) const`
- `template<>`  
`size_t operator() (long long __val) const`
- `template<>`  
`size_t operator() (long __val) const`
- `template<>`  
`size_t operator() (int __val) const`
- `template<>`  
`size_t operator() (short __val) const`
- `template<>`  
`size_t operator() (char32_t __val) const`
- `template<>`  
`size_t operator() (char16_t __val) const`

- `template<>`  
`size_t operator() (wchar_t __val) const`
- `template<>`  
`size_t operator() (unsigned char __val) const`
- `template<>`  
`size_t operator() (signed char __val) const`
- `template<>`  
`size_t operator() (char __val) const`
- `template<>`  
`size_t operator() (bool __val) const`

#### 5.491.1 Detailed Description

`template<typename _Tp> struct std::hash< _Tp >`

Primary class template hash.

Definition at line 58 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

## 5.492 `std::hash< __debug::bitset< _Nb > >` Struct Template Reference

[std::hash](#) specialization for `bitset`.

Inherits `std::__hash_base< size_t, __debug::bitset< _Nb > >`.

#### Public Types

- `typedef _Arg argument_type`
- `typedef _Result result_type`

#### Public Member Functions

- `size_t operator() (const \_\_debug::bitset< \_Nb > &__b) const`

#### 5.492.1 Detailed Description

`template<size_t _Nb> struct std::hash< \_\_debug::bitset< \_Nb > >`

[std::hash](#) specialization for `bitset`.

Definition at line 408 of file `debug/bitset`.

The documentation for this struct was generated from the following file:

- [debug/bitset](#)

### **5.493 `std::hash< __debug::vector< bool, _Alloc > >` Struct Template Reference**

[std::hash](#) specialization for `vector<bool>`.

Inherits `std::__hash_base< size_t, __debug::vector< bool, _Alloc > >`.

#### **Public Types**

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

#### **Public Member Functions**

- `size_t operator() (const \_\_debug::vector< bool, \_Alloc > &__b) const`

#### **5.493.1 Detailed Description**

**template<typename \_Alloc> struct `std::hash< __debug::vector< bool, _Alloc >`**  
**>**

[std::hash](#) specialization for `vector<bool>`.

Definition at line 593 of file `debug/vector`.

The documentation for this struct was generated from the following file:

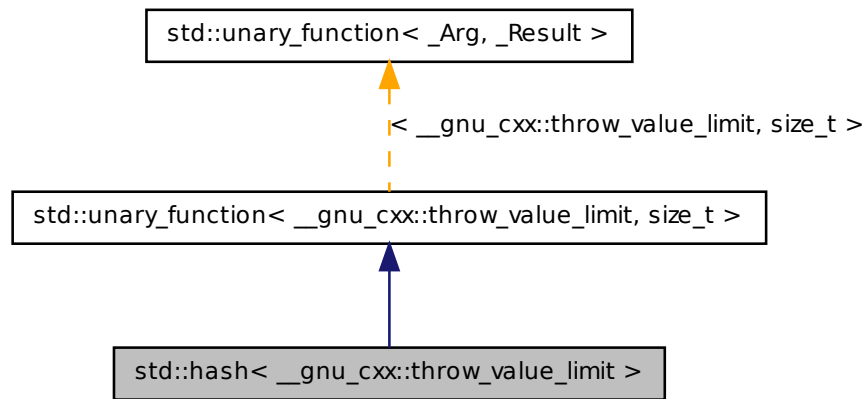
- [debug/vector](#)

### **5.494 `std::hash< __gnu_cxx::throw_value_limit >` Struct Template Reference**

Explicit specialization of [std::hash](#) for `__gnu_cxx::throw_value_limit`.

## 5.494 `std::hash< __gnu_cxx::throw_value_limit >` Struct Template Reference

Inheritance diagram for `std::hash< __gnu_cxx::throw_value_limit >`:



### Public Types

- typedef `__gnu_cxx::throw_value_limit` `argument_type`
- typedef `size_t` `result_type`

### Public Member Functions

- `size_t operator() (const __gnu_cxx::throw_value_limit &__val) const`

#### 5.494.1 Detailed Description

**template<> struct `std::hash< __gnu_cxx::throw_value_limit >`**

Explicit specialization of `std::hash` for `__gnu_cxx::throw_value_limit`.

Definition at line 737 of file `throw_allocator.h`.

## 5.494.2 Member Typedef Documentation

**5.494.2.1** `typedef __gnu_cxx::throw_value_limit std::unary_function< __gnu_cxx::throw_value_limit, size_t >::argument_type [inherited]`

`argument_type` is the type of the argument

Definition at line 105 of file `stl_function.h`.

**5.494.2.2** `typedef size_t std::unary_function< __gnu_cxx::throw_value_limit, size_t >::result_type [inherited]`

`result_type` is the return type

Definition at line 108 of file `stl_function.h`.

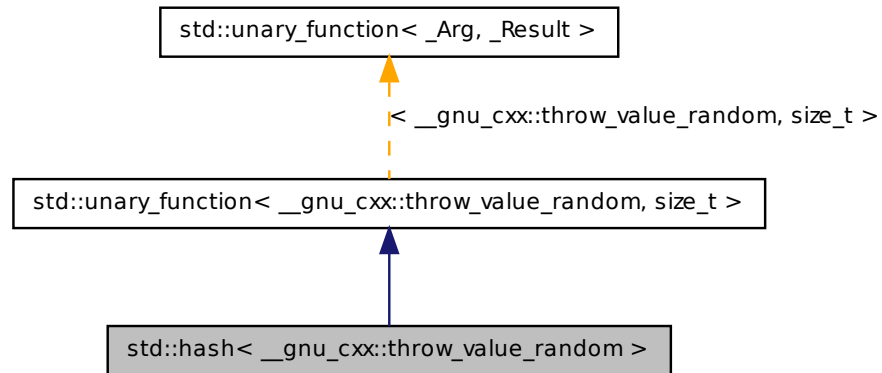
The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.495 `std::hash< __gnu_cxx::throw_value_random >` Struct Template Reference

Explicit specialization of [std::hash](#) for [\\_\\_gnu\\_cxx::throw\\_value\\_limit](#).

Inheritance diagram for `std::hash< __gnu_cxx::throw_value_random >`:



### Public Types

- typedef `__gnu_cxx::throw_value_random` `argument_type`
- typedef `size_t` `result_type`

### Public Member Functions

- `size_t operator() (const __gnu_cxx::throw_value_random &__val) const`

#### 5.495.1 Detailed Description

**template<> struct `std::hash< __gnu_cxx::throw_value_random >`**

Explicit specialization of `std::hash` for `__gnu_cxx::throw_value_limit`.

Definition at line 751 of file `throw_allocator.h`.

## 5.495.2 Member Typedef Documentation

**5.495.2.1** `typedef __gnu_cxx::throw_value_random std::unary_function< __gnu_cxx::throw_value_random , size_t >::argument_type`  
[`inherited`]

`argument_type` is the type of the argument

Definition at line 105 of file `stl_function.h`.

**5.495.2.2** `typedef size_t std::unary_function< __gnu_cxx::throw_value_random , size_t >::result_type`  
[`inherited`]

`result_type` is the return type

Definition at line 108 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 5.496 `std::hash< __profile::bitset< _Nb > >` Struct Template Reference

[std::hash](#) specialization for `bitset`.

Inherits `std::__hash_base< size_t, __profile::bitset< _Nb > >`.

### Public Types

- `typedef _Arg argument_type`
- `typedef _Result result_type`

### Public Member Functions

- `size_t operator() (const \_\_profile::bitset< _Nb > &__b) const`

### 5.496.1 Detailed Description

**template<size\_t \_Nb> struct std::hash< \_\_profile::bitset< \_Nb > >**

[std::hash](#) specialization for `bitset`.

Definition at line 366 of file `profile/bitset`.

The documentation for this struct was generated from the following file:

- [profile/bitset](#)

## 5.497 `std::hash< __profile::vector< bool, _Alloc > >` Struct Template Reference

[std::hash](#) specialization for `vector<bool>`.

Inherits `std::__hash_base< size_t, __profile::vector< bool, _Alloc > >`.

### Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

### Public Member Functions

- `size_t operator() (const __profile::vector< bool, _Alloc > &__b) const`

### 5.497.1 Detailed Description

**template<typename \_Alloc> struct std::hash< \_\_profile::vector< bool, \_Alloc > >**

[std::hash](#) specialization for `vector<bool>`.

Definition at line 507 of file `profile/vector`.

The documentation for this struct was generated from the following file:

- [profile/vector](#)

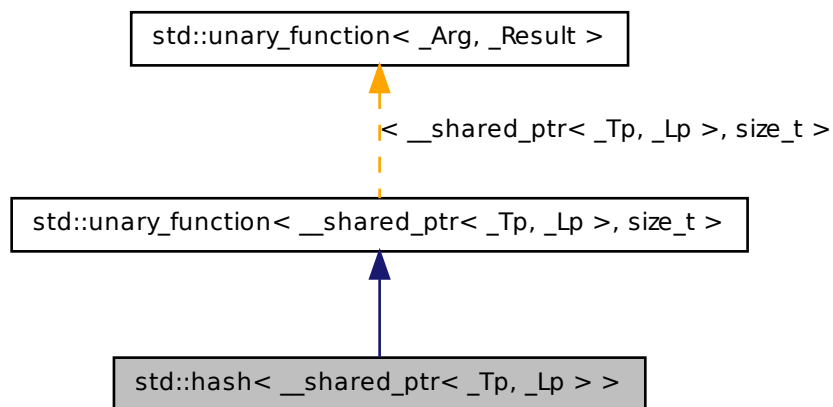
## 5.498 `std::hash< __shared_ptr< _Tp, _Lp > >` Struct Template Reference

[std::hash](#) specialization for `__shared_ptr`.



## 5.498 `std::hash< __shared_ptr< _Tp, _Lp > >` Struct Template Reference 2710

Inheritance diagram for `std::hash< __shared_ptr< _Tp, _Lp > >`:



### Public Types

- typedef `__shared_ptr< _Tp, _Lp >` [argument\\_type](#)
- typedef `size_t` [result\\_type](#)

### Public Member Functions

- `size_t operator() (const __shared_ptr< _Tp, _Lp > &__s) const`

#### 5.498.1 Detailed Description

**template<typename \_Tp, \_Lock\_policy \_Lp> struct `std::hash< __shared_ptr< _Tp, _Lp > >`**

[std::hash](#) specialization for `__shared_ptr`.

Definition at line 1373 of file `shared_ptr_base.h`.

### 5.498.2 Member Typedef Documentation

**5.498.2.1** `typedef __shared_ptr< _Tp, _Lp > std::unary_function< __shared_ptr< _Tp, _Lp >, size_t >::argument_type [inherited]`

`argument_type` is the type of the argument

Definition at line 105 of file `stl_function.h`.

**5.498.2.2** `typedef size_t std::unary_function< __shared_ptr< _Tp, _Lp >, size_t >::result_type [inherited]`

`result_type` is the return type

Definition at line 108 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [shared\\_ptr\\_base.h](#)

## 5.499 `std::hash< _Tp * >` Struct Template Reference

Partial specializations for pointer types.

Inherits `std::__hash_base< size_t, _Tp * >`.

### Public Types

- `typedef _Arg argument_type`
- `typedef _Result result_type`

### Public Member Functions

- `size_t operator() (_Tp *__p) const`

### 5.499.1 Detailed Description

`template<typename _Tp> struct std::hash< _Tp * >`

Partial specializations for pointer types.

Definition at line 66 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

## 5.500 `std::hash< error_code >` Struct Template Reference

`std::hash` specialization for `error_code`.

Inherits `std::__hash_base< size_t, error_code >`.

### Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

### Public Member Functions

- `size_t operator() (const error_code &__e) const`

#### 5.500.1 Detailed Description

`template<> struct std::hash< error_code >`

`std::hash` specialization for `error_code`.

Definition at line 360 of file `system_error`.

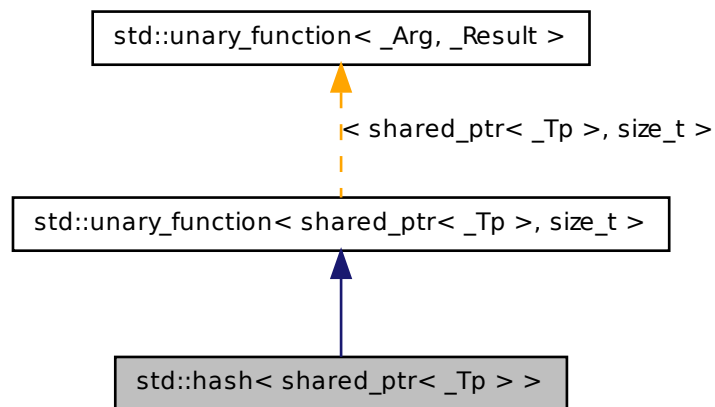
The documentation for this struct was generated from the following file:

- [system\\_error](#)

## 5.501 `std::hash< shared_ptr< _Tp > >` Struct Template Reference

`std::hash` specialization for `shared_ptr`.

Inheritance diagram for `std::hash< shared_ptr< _Tp > >`:



### Public Types

- typedef `shared_ptr< _Tp >` `argument_type`
- typedef `size_t` `result_type`

### Public Member Functions

- `size_t operator() (const shared_ptr< _Tp > &__s) const`

#### 5.501.1 Detailed Description

`template<typename _Tp> struct std::hash< shared_ptr< _Tp > >`

`std::hash` specialization for `shared_ptr`.

Definition at line 552 of file `shared_ptr.h`.

### 5.501.2 Member Typedef Documentation

**5.501.2.1** `typedef shared_ptr< _Tp > std::unary_function< shared_ptr< _Tp >, size_t >::argument_type [inherited]`

`argument_type` is the type of the argument

Definition at line 105 of file `stl_function.h`.

**5.501.2.2** `typedef size_t std::unary_function< shared_ptr< _Tp >, size_t >::result_type [inherited]`

`result_type` is the return type

Definition at line 108 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [shared\\_ptr.h](#)

## 5.502 `std::hash< string >` Struct Template Reference

[std::hash](#) specialization for string.

Inherits `std::__hash_base< size_t, string >`.

### Public Types

- `typedef _Arg argument_type`
- `typedef _Result result_type`

### Public Member Functions

- `size_t operator() (const string &__s) const`

### 5.502.1 Detailed Description

`template<> struct std::hash< string >`

[std::hash](#) specialization for string.

Definition at line 2985 of file `basic_string.h`.

The documentation for this struct was generated from the following file:

- [basic\\_string.h](#)

### 5.503 `std::hash< thread::id >` Struct Template Reference

[std::hash](#) specialization for [thread::id](#).

Inherits `std::__hash_base< size_t, thread::id >`.

#### Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

#### Public Member Functions

- `size_t operator()` (const [thread::id](#) &\_\_id) const

#### 5.503.1 Detailed Description

`template<> struct std::hash< thread::id >`

[std::hash](#) specialization for [thread::id](#).

Definition at line 223 of file `thread`.

The documentation for this struct was generated from the following file:

- [thread](#)

### 5.504 `std::hash< type_index >` Struct Template Reference

[std::hash](#) specialization for [type\\_index](#).

#### Public Types

- typedef [type\\_index](#) **argument\_type**
- typedef `size_t` **result\_type**

#### Public Member Functions

- `size_t operator()` (const [type\\_index](#) &\_\_ti) const

#### 5.504.1 Detailed Description

`template<> struct std::hash< type_index >`

[std::hash](#) specialization for [type\\_index](#).

Definition at line 94 of file `typeindex`.

The documentation for this struct was generated from the following file:

- [typeindex](#)

### 5.505 `std::hash< u16string >` Struct Template Reference

[std::hash](#) specialization for `u16string`.

Inherits `std::__hash_base< size_t, u16string >`.

#### Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

#### Public Member Functions

- `size_t operator() (const u16string &__s) const`

#### 5.505.1 Detailed Description

`template<> struct std::hash< u16string >`

[std::hash](#) specialization for `u16string`.

Definition at line 3010 of file `basic_string.h`.

The documentation for this struct was generated from the following file:

- [basic\\_string.h](#)

### 5.506 `std::hash< u32string >` Struct Template Reference

[std::hash](#) specialization for `u32string`.

Inherits `std::__hash_base< size_t, u32string >`.

### Public Types

- typedef \_Arg **argument\_type**
- typedef \_Result **result\_type**

### Public Member Functions

- size\_t **operator()** (const [u32string](#) &\_\_s) const

#### 5.506.1 Detailed Description

**template<> struct std::hash< u32string >**

[std::hash](#) specialization for [u32string](#).

Definition at line 3021 of file [basic\\_string.h](#).

The documentation for this struct was generated from the following file:

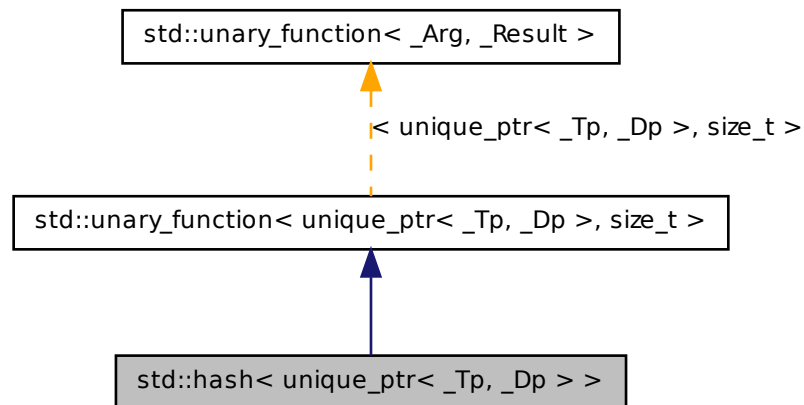
- [basic\\_string.h](#)

### 5.507 std::hash< unique\_ptr< \_Tp, \_Dp > > Struct Template Reference

[std::hash](#) specialization for [unique\\_ptr](#).



Inheritance diagram for `std::hash< unique_ptr< _Tp, _Dp > >`:



### Public Types

- typedef `unique_ptr< _Tp, _Dp >` `argument_type`
- typedef `size_t` `result_type`

### Public Member Functions

- `size_t operator() (const unique_ptr< _Tp, _Dp > &__u) const`

#### 5.507.1 Detailed Description

`template<typename _Tp, typename _Dp> struct std::hash< unique_ptr< _Tp, _Dp > >`

`std::hash` specialization for `unique_ptr`.

Definition at line 493 of file `unique_ptr.h`.

### 5.507.2 Member Typedef Documentation

**5.507.2.1** `typedef unique_ptr< _Tp, _Dp > std::unary_function< unique_ptr< _Tp, _Dp >, size_t >::argument_type [inherited]`

`argument_type` is the type of the argument

Definition at line 105 of file `stl_function.h`.

**5.507.2.2** `typedef size_t std::unary_function< unique_ptr< _Tp, _Dp >, size_t >::result_type [inherited]`

`result_type` is the return type

Definition at line 108 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [unique\\_ptr.h](#)

## 5.508 `std::hash< wstring >` Struct Template Reference

[std::hash](#) specialization for `wstring`.

Inherits `std::__hash_base< size_t, wstring >`.

### Public Types

- `typedef _Arg argument_type`
- `typedef _Result result_type`

### Public Member Functions

- `size_t operator() (const wstring &__s) const`

### 5.508.1 Detailed Description

`template<> struct std::hash< wstring >`

[std::hash](#) specialization for `wstring`.

Definition at line 2996 of file `basic_string.h`.

The documentation for this struct was generated from the following file:

- [basic\\_string.h](#)

## 5.509 `std::hash<::bitset< _Nb > >` Struct Template Reference

[std::hash](#) specialization for `bitset`.

Inherits `std::__hash_base< size_t,::bitset< _Nb > >`.

### Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

### Public Member Functions

- `size_t operator() (const ::bitset< _Nb > &__b) const`

#### 5.509.1 Detailed Description

`template<size_t _Nb> struct std::hash<::bitset< _Nb > >`

[std::hash](#) specialization for `bitset`.

Definition at line 1524 of file `bitset`.

The documentation for this struct was generated from the following file:

- [bitset](#)

## 5.510 `std::hash<::vector< bool, _Alloc > >` Struct Template Reference

[std::hash](#) specialization for `vector<bool>`.

Inherits `std::__hash_base< size_t,::vector< bool, _Alloc > >`.

### Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

## Public Member Functions

- `size_t operator() (const ::vector< bool, _Alloc > &__b) const`

### 5.510.1 Detailed Description

**template<typename \_Alloc> struct std::hash<::vector< bool, \_Alloc > >**

[std::hash](#) specialization for `vector<bool>`.

Definition at line 1048 of file `stl_bvector.h`.

The documentation for this struct was generated from the following files:

- [stl\\_bvector.h](#)
- [vector.tcc](#)

## **5.511 std::independent\_bits\_engine< \_RandomNumberEngine, \_\_w, \_UIntType > Class Template Reference**

### Public Types

- `typedef _UIntType result\_type`

### Public Member Functions

- [independent\\_bits\\_engine](#) ()
- [independent\\_bits\\_engine](#) (const \_RandomNumberEngine &\_\_rne)
- [independent\\_bits\\_engine](#) ([result\\_type](#) \_\_s)
- `template<typename _Sseq , typename = typename std::enable_if<!std::is_same<_Sseq, independent_bits_engine>::value && !std::is_same<_Sseq, _RandomNumberEngine>::value>::type>`  
[independent\\_bits\\_engine](#) (\_Sseq &\_\_q)
- [independent\\_bits\\_engine](#) (\_RandomNumberEngine &&\_\_rne)
- `const _RandomNumberEngine & base () const`
- `void discard (unsigned long long __z)`
- [result\\_type operator\(\)](#) ()
- `void seed ()`
- `template<typename _Sseq >`  
`void seed (_Sseq &__q)`
- `void seed (result\_type __s)`

## Static Public Member Functions

- static constexpr [result\\_type](#) max ()
- static constexpr [result\\_type](#) min ()

## Friends

- bool [operator==](#) (const [independent\\_bits\\_engine](#) &\_\_lhs, const [independent\\_bits\\_engine](#) &\_\_rhs)
- template<typename \_CharT, typename \_Traits >  
[std::basic\\_istream](#)< \_CharT, \_Traits > & [operator>>](#) ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, [std::independent\\_bits\\_engine](#)< \_RandomNumberEngine, \_\_w, \_UIntType > &\_\_x)

### 5.511.1 Detailed Description

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
class std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType
>
```

Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits \_\_w.

Definition at line 999 of file random.h.

### 5.511.2 Member Typedef Documentation

5.511.2.1 template<typename \_RandomNumberEngine, size\_t \_\_w, typename \_UIntType> typedef \_UIntType std::independent\_bits\_engine<\_RandomNumberEngine, \_\_w, \_UIntType>::result\_type

The type of the generated random value.

Definition at line 1008 of file random.h.

### 5.511.3 Constructor & Destructor Documentation

5.511.3.1 template<typename \_RandomNumberEngine, size\_t \_\_w, typename \_UIntType> std::independent\_bits\_engine<\_RandomNumberEngine, \_\_w, \_UIntType>::independent\_bits\_engine ( ) [inline]

Constructs a default independent\_bits\_engine engine.

The underlying engine is default constructed as well.

Definition at line 1015 of file random.h.

**5.511.3.2** `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::independent_bits_engine ( const _RandomNumberEngine & __rne ) [inline, explicit]`

Copy constructs a independent\_bits\_engine engine.

Copies an existing base class random number generator.

#### Parameters

*rng* An existing (base class) engine object.

Definition at line 1025 of file random.h.

**5.511.3.3** `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::independent_bits_engine ( _RandomNumberEngine && __rne ) [inline, explicit]`

Move constructs a independent\_bits\_engine engine.

Copies an existing base class random number generator.

#### Parameters

*rng* An existing (base class) engine object.

Definition at line 1035 of file random.h.

**5.511.3.4** `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::independent_bits_engine ( result_type __s ) [inline, explicit]`

Seed constructs a independent\_bits\_engine engine.

Constructs the underlying generator engine seeded with \_\_s.

**Parameters**

\_\_s A seed value for the base class engine.

Definition at line 1045 of file random.h.

```
5.511.3.5 template<typename _RandomNumberEngine, size_t
__w, typename _UIntType> template<typename _Sseq ,
typename = typename std::enable_if<!std::is_same<_Sseq,
independent_bits_engine>::value && !std::is_same<_Sseq,
RandomNumberEngine>::value> ::type> std::independent
bits_engine< _RandomNumberEngine, __w, _UIntType
>::independent_bits_engine (_Sseq & __q) [inline,
explicit]
```

Generator construct a independent\_bits\_engine engine.

**Parameters**

\_\_q A seed sequence.

Definition at line 1058 of file random.h.

**5.511.4 Member Function Documentation**

```
5.511.4.1 template<typename _RandomNumberEngine, size_t __w,
typename _UIntType> const _RandomNumberEngine&
std::independent_bits_engine< _RandomNumberEngine, __w,
 UIntType >::base () const [inline]
```

Gets a const reference to the underlying generator engine object.

Definition at line 1093 of file random.h.

```
5.511.4.2 template<typename _RandomNumberEngine, size_t __w,
typename _UIntType> void std::independent_bits_engine<
_RandomNumberEngine, __w, _UIntType >::discard (unsigned
long long __z) [inline]
```

Discard a sequence of random numbers.

Definition at line 1114 of file random.h.

**5.511.4.3** `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> static constexpr result_type std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::max( ) [inline, static]`

Gets the maximum value in the generated random number range.

Definition at line 1107 of file random.h.

**5.511.4.4** `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> static constexpr result_type std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::min( ) [inline, static]`

Gets the minimum value in the generated random number range.

Definition at line 1100 of file random.h.

**5.511.4.5** `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::result_type std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::operator()( )`

Gets the next value in the generated random number sequence.

Definition at line 727 of file random.tcc.

References std::log().

**5.511.4.6** `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> template<typename _Sseq> void std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::seed( _Sseq & __q ) [inline]`

Reseeds the independent\_bits\_engine object with the given seed sequence.



### Parameters

**\_\_q** A seed generator function.

Definition at line 1085 of file random.h.

```
5.511.4.7 template<typename _RandomNumberEngine, size_t __w,
typename _UIntType> void std::independent_bits_engine<
_RandomNumberEngine, __w, _UIntType >::seed (result_type __s
) [inline]
```

Reseeds the independent\_bits\_engine object with the default seed for the underlying base class generator engine.

Definition at line 1075 of file random.h.

```
5.511.4.8 template<typename _RandomNumberEngine, size_t __w,
typename _UIntType> void std::independent_bits_engine<
_RandomNumberEngine, __w, _UIntType >::seed () [inline]
```

Reseeds the independent\_bits\_engine object with the default seed for the underlying base class generator engine.

Definition at line 1067 of file random.h.

### 5.511.5 Friends And Related Function Documentation

```
5.511.5.1 template<typename _RandomNumberEngine, size_t __w, typename
_UIntType> bool operator== (const independent_bits_engine<
_RandomNumberEngine, __w, _UIntType > & __lhs, const
independent_bits_engine<_RandomNumberEngine, __w, _UIntType
> & __rhs) [friend]
```

Compares two independent\_bits\_engine random number generator objects of the same type for equality.

### Parameters

**\_\_lhs** A independent\_bits\_engine random number generator object.

**\_\_rhs** Another independent\_bits\_engine random number generator object.

**Returns**

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 1139 of file random.h.

**5.511.5.2** `template<typename _RandomNumberEngine, size_t __w,  
typename _UIntType> template<typename _CharT ,  
typename _Traits > std::basic_istream<_CharT, _Traits>&  
operator>> ( std::basic_istream<_CharT, _Traits> & __is,  
std::independent_bits_engine<_RandomNumberEngine, __w,  
_UIntType> & __x ) [friend]`

Extracts the current state of a % subtract\_with\_carry\_engine random number generator engine \_\_x from the input stream \_\_is.

**Parameters**

`__is` An input stream.

`__x` A independent\_bits\_engine random number generator engine.

**Returns**

The input stream with the state of \_\_x extracted or in an error state.

Definition at line 1157 of file random.h.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

**5.512 `std::indirect_array<_Tp>` Class Template Reference**

Reference to arbitrary subset of an array.

**Public Types**

- `typedef _Tp value_type`

**Public Member Functions**

- [indirect\\_array](#) (const [indirect\\_array](#) &)
- template<class \_Dom >  
void **operator**%= (const \_Expr< \_Dom, \_Tp > &) const
- void **operator**%= (const [valarray](#)< \_Tp > &) const
- void **operator**&= (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void **operator**&= (const \_Expr< \_Dom, \_Tp > &) const
- template<class \_Dom >  
void **operator**\*= (const \_Expr< \_Dom, \_Tp > &) const
- void **operator**\*= (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void **operator**+= (const \_Expr< \_Dom, \_Tp > &) const
- void **operator**+= (const [valarray](#)< \_Tp > &) const
- void **operator**-= (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void **operator**-= (const \_Expr< \_Dom, \_Tp > &) const
- template<class \_Dom >  
void **operator**/= (const \_Expr< \_Dom, \_Tp > &) const
- void **operator**/= (const [valarray](#)< \_Tp > &) const
- void **operator**<<= (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void **operator**<<= (const \_Expr< \_Dom, \_Tp > &) const
- void **operator**= (const \_Tp &) const
- [indirect\\_array](#) & **operator**= (const [indirect\\_array](#) &)
- template<class \_Dom >  
void **operator**= (const \_Expr< \_Dom, \_Tp > &) const
- void **operator**= (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void **operator**>>= (const \_Expr< \_Dom, \_Tp > &) const
- void **operator**>>= (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void **operator**^= (const \_Expr< \_Dom, \_Tp > &) const
- void **operator**^= (const [valarray](#)< \_Tp > &) const
- void **operator**|= (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void **operator**|= (const \_Expr< \_Dom, \_Tp > &) const

**Friends**

- class [gslice\\_array](#)< \_Tp >
- class [valarray](#)< \_Tp >

### 5.512.1 Detailed Description

**template<class \_Tp> class std::indirect\_array< \_Tp >**

Reference to arbitrary subset of an array. An [indirect\\_array](#) is a reference to the actual elements of an array specified by an ordered array of indices. The way to get an [indirect\\_array](#) is to call operator[](valarray<size\_t>) on a valarray. The returned [indirect\\_array](#) then permits carrying operations out on the referenced subset of elements in the original valarray.

For example, if an [indirect\\_array](#) is obtained using the array (4,2,0) as an argument, and then assigned to an array containing (1,2,3), then the underlying array will have array[0]==3, array[2]==2, and array[4]==1.

#### Parameters

*Tp* Element type.

Definition at line 63 of file indirect\_array.h.

### 5.512.2 Member Function Documentation

**5.512.2.1    template<class \_Tp> void std::indirect\_array< \_Tp >::operator%=(  
              ( const valarray< \_Tp > & ) const**

Modulo slice elements by corresponding elements of *v*.

**5.512.2.2    template<class \_Tp> void std::indirect\_array< \_Tp >::operator&=(  
              ( const valarray< \_Tp > & ) const**

Logical and slice elements with corresponding elements of *v*.

**5.512.2.3    template<class \_Tp> void std::indirect\_array< \_Tp >::operator\*=(  
              const valarray< \_Tp > & ) const**

Multiply slice elements by corresponding elements of *v*.

**5.512.2.4** `template<class _Tp> void std::indirect_array<_Tp>::operator+=(  
const valarray<_Tp> & ) const`

Add corresponding elements of *v* to slice elements.

**5.512.2.5** `template<class _Tp> void std::indirect_array<_Tp>::operator-=(  
const valarray<_Tp> & ) const`

Subtract corresponding elements of *v* from slice elements.

**5.512.2.6** `template<class _Tp> void std::indirect_array<_Tp>::operator/=(  
const valarray<_Tp> & ) const`

Divide slice elements by corresponding elements of *v*.

**5.512.2.7** `template<class _Tp> void std::indirect_array<_Tp>  
>::operator<<=( const valarray<_Tp> & ) const`

Left shift slice elements by corresponding elements of *v*.

**5.512.2.8** `template<class _Tp> void std::indirect_array<_Tp>  
>::operator>>=( const valarray<_Tp> & ) const`

Right shift slice elements by corresponding elements of *v*.

**5.512.2.9** `template<class _Tp> void std::indirect_array<_Tp>::operator^=(  
const valarray<_Tp> & ) const`

Logical xor slice elements with corresponding elements of *v*.

**5.512.2.10** `template<class _Tp> void std::indirect_array<_Tp>::operator|=`  
`( const valarray<_Tp> & ) const`

Logical or slice elements with corresponding elements of *v*.

The documentation for this class was generated from the following file:

- [indirect\\_array.h](#)

## 5.513 `std::initializer_list<_E>` Class Template Reference

[initializer\\_list](#)

### Public Types

- `typedef const _E * const_iterator`
- `typedef const _E & const_reference`
- `typedef const _E * iterator`
- `typedef const _E & reference`
- `typedef size_t size_type`
- `typedef _E value_type`

### Public Member Functions

- `constexpr const_iterator begin ()`
- `constexpr const_iterator end ()`
- `constexpr size_type size ()`

#### 5.513.1 Detailed Description

`template<class _E> class std::initializer_list<_E>`

[initializer\\_list](#)

Definition at line 45 of file `initializer_list`.

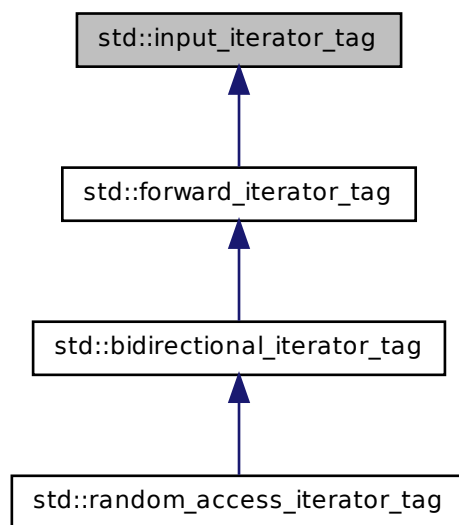
The documentation for this class was generated from the following file:

- [initializer\\_list](#)

## 5.514 `std::input_iterator_tag` Struct Reference

Marking input iterators.

Inheritance diagram for `std::input_iterator_tag`:



### 5.514.1 Detailed Description

Marking input iterators.

Definition at line 90 of file `stl_iterator_base_types.h`.

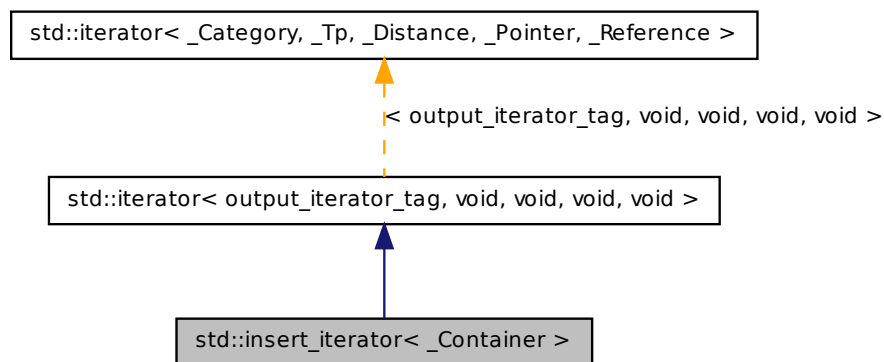
The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 5.515 `std::insert_iterator< _Container >` Class Template Reference

Turns assignment into insertion.

Inheritance diagram for `std::insert_iterator< _Container >`:



### Public Types

- `typedef _Container container_type`
- `typedef void difference_type`
- `typedef output_iterator_tag iterator_category`
- `typedef void pointer`
- `typedef void reference`
- `typedef void value_type`

### Public Member Functions

- `insert_iterator` (`_Container &__x`, `typename _Container::iterator __i`)
- `insert_iterator & operator* ()`
- `insert_iterator & operator++ ()`
- `insert_iterator & operator++ (int)`
- `insert_iterator & operator= (const typename _Container::value_type &__value)`
- `insert_iterator & operator= (typename _Container::value_type &&__value)`

### Protected Attributes

- `_Container * container`
- `_Container::iterator iter`



### 5.515.1 Detailed Description

**template<typename \_Container> class std::insert\_iterator< \_Container >**

Turns assignment into insertion. These are output iterators, constructed from a container-of-T. Assigning a T to the iterator inserts it in the container at the iterator's position, rather than overwriting the value at that position.

(Sequences will actually insert a *copy* of the value before the iterator's position.)

Tip: Using the inserter function to create these iterators can save typing.

Definition at line 581 of file stl\_iterator.h.

### 5.515.2 Member Typedef Documentation

**5.515.2.1 template<typename \_Container> typedef \_Container  
std::insert\_iterator< \_Container >::container\_type**

A nested typedef for the type of whatever container you used.

Definition at line 590 of file stl\_iterator.h.

**5.515.2.2 typedef void std::iterator< output\_iterator\_tag , void , void , void ,  
void >::difference\_type [inherited]**

Distance between iterators is represented as this type.

Definition at line 126 of file stl\_iterator\_base\_types.h.

**5.515.2.3 typedef output\_iterator\_tag std::iterator< output\_iterator\_tag , void  
, void , void , void >::iterator\_category [inherited]**

One of the [tag types](#).

Definition at line 122 of file stl\_iterator\_base\_types.h.

**5.515.2.4 typedef void std::iterator< output\_iterator\_tag , void , void , void ,  
void >::pointer [inherited]**

This type represents a pointer-to-value\_type.

Definition at line 128 of file stl\_iterator\_base\_types.h.

**5.515.2.5** `typedef void std::iterator< output_iterator_tag , void , void , void ,  
void >::reference [inherited]`

This type represents a reference-to-value\_type.

Definition at line 130 of file stl\_iterator\_base\_types.h.

**5.515.2.6** `typedef void std::iterator< output_iterator_tag , void , void , void ,  
void >::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 124 of file stl\_iterator\_base\_types.h.

### **5.515.3 Constructor & Destructor Documentation**

**5.515.3.1** `template<typename _Container> std::insert_iterator<  
_Container >::insert_iterator ( _Container & __x, typename  
_Container::iterator __i ) [inline]`

The only way to create this iterator is with a container and an initial position (a normal iterator into the container).

Definition at line 596 of file stl\_iterator.h.

### **5.515.4 Member Function Documentation**

**5.515.4.1** `template<typename _Container> insert_iterator&  
std::insert_iterator< _Container >::operator*( ) [inline]`

Simply returns \*this.

Definition at line 650 of file stl\_iterator.h.

**5.515.4.2** `template<typename _Container> insert_iterator&  
std::insert_iterator<_Container>::operator++( int ) [inline]`

Simply returns `*this`. (This iterator does not *move*.).

Definition at line 660 of file `stl_iterator.h`.

**5.515.4.3** `template<typename _Container> insert_iterator&  
std::insert_iterator<_Container>::operator++( ) [inline]`

Simply returns `*this`. (This iterator does not *move*.).

Definition at line 655 of file `stl_iterator.h`.

**5.515.4.4** `template<typename _Container> insert_iterator&  
std::insert_iterator<_Container>::operator=( const typename  
_Container::value_type & __value ) [inline]`

### Parameters

**value** An instance of whatever type `container_type::const_reference` is; presumably a reference-to-const `T` for `container<T>`.

### Returns

This iterator, for chained operations.

This kind of iterator maintains its own position in the container. Assigning a value to the iterator will insert the value into the container at the place before the iterator.

The position is maintained such that subsequent assignments will insert values immediately after one another. For example,

```
// vector v contains A and Z

insert_iterator i (v, ++v.begin());
i = 1;
i = 2;
i = 3;

// vector v contains A, 1, 2, 3, and Z
```

Definition at line 632 of file `stl_iterator.h`.

The documentation for this class was generated from the following file:

- [stl\\_iterator.h](#)

## 5.516 `std::integral_constant< _Tp, __v >` Struct Template Reference

### [integral\\_constant](#)

Inherited by `std::__is_floating_point_helper< typename >`, `std::__is_integral_helper< typename >`, `std::__is_member_function_pointer_helper< typename >`, `std::__is_member_object_pointer_helper< typename >`, [std::\\_\\_is\\_member\\_pointer\\_helper< \\_Tp >](#), `std::__is_nullptr_t_helper< typename >`, `std::__is_pointer_helper< typename >`, `std::__is_signed_helper< _Tp, bool, bool >`, `std::__is_void_helper< typename >`, `std::__uses_allocator_helper< _Tp, _Alloc, bool >`, `std::chrono::__is_duration< _Tp >`, `std::chrono::__is_duration< duration< _Rep, _Period > >`, `std::chrono::__is_ratio< T >`, [std::is\\_array< typename >](#), `std::is_array< _Tp[_Size]>`, [std::is\\_bind\\_expression< \\_Tp >](#), [std::is\\_bind\\_expression< \\_Bind< \\_Signature > >](#), [std::is\\_bind\\_expression< \\_Bind\\_result< \\_Result, \\_Signature > >](#), [std::is\\_const< typename >](#), [std::is\\_error\\_code\\_enum< \\_Tp >](#), [std::is\\_error\\_code\\_enum< future\\_errc >](#), [std::is\\_error\\_condition\\_enum< \\_Tp >](#), [std::is\\_function< typename >](#), `std::is_function< _Res(_ArgTypes...) const >`, `std::is_function< _Res(_ArgTypes...) const volatile >`, `std::is_function< _Res(_ArgTypes...) >`, `std::is_function< _Res(_ArgTypes.....) const >`, `std::is_function< _Res(_ArgTypes.....) const volatile >`, `std::is_function< _Res(_ArgTypes.....) volatile >`, [std::is\\_lvalue\\_reference< typename >](#), [std::is\\_rvalue\\_reference< typename >](#), `std::is_rvalue_reference< _Tp && >`, [std::is\\_same< typename, typename >](#), [std::is\\_volatile< typename >](#), `std::uses_allocator< priority_queue< _Tp, _Sequence, _Compare >, _Alloc >`, `std::uses_allocator< queue< _Tp, _Seq >, _Alloc >`, and `std::uses_allocator< stack< _Tp, _Seq >, _Alloc >`.

### Public Types

- typedef [integral\\_constant< \\_Tp, \\_\\_v >](#) **type**
- typedef `_Tp` **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr `_Tp` **value**

### 5.516.1 Detailed Description

`template<typename _Tp, _Tp __v> struct std::integral_constant< _Tp, __v >`

[integral\\_constant](#)

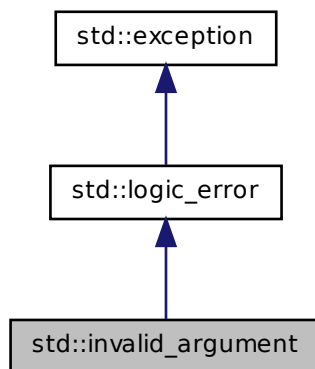
Definition at line 72 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.517 std::invalid\_argument Class Reference

Inheritance diagram for `std::invalid_argument`:



### Public Member Functions

- **invalid\_argument** (const [string](#) &\_\_arg)
- virtual const char \* [what](#) () const throw ()

### 5.517.1 Detailed Description

Thrown to report invalid arguments to functions.

Definition at line 83 of file stdexcept.

## 5.517.2 Member Function Documentation

### 5.517.2.1 virtual const char\* std::logic\_error::what ( ) const throw () [virtual, inherited]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future\\_error](#).

The documentation for this class was generated from the following file:

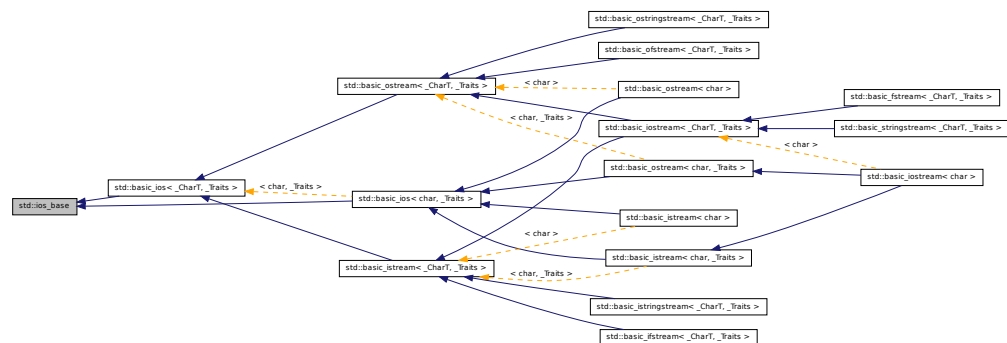
- [stdexcept](#)

## 5.518 std::ios\_base Class Reference

The base of the I/O class hierarchy.

This class defines everything that can be defined about I/O that does not depend on the type of characters being input or output. Most people will only see [ios\\_base](#) when they need to specify the full name of the various I/O flags (e.g., the openmodes).

Inheritance diagram for `std::ios_base`:



## Classes

- class [failure](#)

*These are thrown to indicate problems with io.*

*27.4.2.1.1 Class `ios_base::failure`.*

### Public Types

- enum `event` { `erase_event`, `imbue_event`, `copyfmt_event` }
- typedef void(\* `event_callback` )(event, `ios_base` &, int)
- typedef `_Ios_Fmtflags` `fmtflags`
- typedef int `io_state`
- typedef `_Ios_Iostate` `iostate`
- typedef int `open_mode`
- typedef `_Ios_Openmode` `openmode`
- typedef int `seek_dir`
- typedef `_Ios_Seekdir` `seekdir`
- typedef `std::streamoff` `streamoff`
- typedef `std::streampos` `streampos`

### Public Member Functions

- virtual `~ios_base` ()
- const `locale` & `_M_getloc` () const
- `fmtflags` `flags` () const
- `fmtflags` `flags` (`fmtflags` \_\_fmtfl)
- `locale` `getloc` () const
- `locale` `imbue` (const `locale` &\_\_loc) throw ()
- long & `iword` (int \_\_ix)
- `streamsize` `precision` (`streamsize` \_\_prec)
- `streamsize` `precision` () const
- void \*& `pword` (int \_\_ix)
- void `register_callback` (`event_callback` \_\_fn, int \_\_index)
- `fmtflags` `setf` (`fmtflags` \_\_fmtfl)
- `fmtflags` `setf` (`fmtflags` \_\_fmtfl, `fmtflags` \_\_mask)
- void `unsetf` (`fmtflags` \_\_mask)
- `streamsize` `width` () const
- `streamsize` `width` (`streamsize` \_\_wide)

### Static Public Member Functions

- static bool `sync_with_stdio` (bool \_\_sync=true)
- static int `xalloc` () throw ()

### Static Public Attributes

- static const `fmtflags` `adjustfield`
- static const `openmode` `app`
- static const `openmode` `ate`
- static const `iostate` `badbit`
- static const `fmtflags` `basefield`
- static const `seekdir` `beg`
- static const `openmode` `binary`
- static const `fmtflags` `boolalpha`
- static const `seekdir` `cur`
- static const `fmtflags` `dec`
- static const `seekdir` `end`
- static const `iostate` `eofbit`
- static const `iostate` `failbit`
- static const `fmtflags` `fixed`
- static const `fmtflags` `floatfield`
- static const `iostate` `goodbit`
- static const `fmtflags` `hex`
- static const `openmode` `in`
- static const `fmtflags` `internal`
- static const `fmtflags` `left`
- static const `fmtflags` `oct`
- static const `openmode` `out`
- static const `fmtflags` `right`
- static const `fmtflags` `scientific`
- static const `fmtflags` `showbase`
- static const `fmtflags` `showpoint`
- static const `fmtflags` `showpos`
- static const `fmtflags` `skipws`
- static const `openmode` `trunc`
- static const `fmtflags` `unitbuf`
- static const `fmtflags` `uppercase`

### Protected Types

- enum { `_S_local_word_size` }

### Protected Member Functions

- void `_M_call_callbacks` (`event` `__ev`) throw ()
- void `_M_dispose_callbacks` (void) throw ()
- `_Words` & `_M_grow_words` (int `__index`, bool `__iword`)
- void `_M_init` () throw ()



### Protected Attributes

- `_Callback_list * _M_callbacks`
- `ios_base::_M_exception`
- `fmtflags _M_flags`
- `locale _M_ios_locale`
- `_Words _M_local_word [_S_local_word_size]`
- `streamsize _M_precision`
- `ios_base::_M_streambuf_state`
- `streamsize _M_width`
- `_Words * _M_word`
- `int _M_word_size`
- `_Words _M_word_zero`

#### 5.518.1 Detailed Description

The base of the I/O class hierarchy.

This class defines everything that can be defined about I/O that does not depend on the type of characters being input or output. Most people will only see `ios_base` when they need to specify the full name of the various I/O flags (e.g., the openmodes).

Definition at line 201 of file `ios_base.h`.

#### 5.518.2 Member Typedef Documentation

##### 5.518.2.1 `typedef void(* std::ios_base::event_callback)(event, ios_base &, int)`

The type of an event callback function.

### Parameters

- event* One of the members of the event enum.
- ios\_base* Reference to the `ios_base` object.
- int* The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 438 of file `ios_base.h`.

### 5.518.2.2 `typedef _Ios_Fmtflags std::ios_base::fmtflags`

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 257 of file `ios_base.h`.

### 5.518.2.3 `typedef _Ios_Iostate std::ios_base::iostate`

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 332 of file `ios_base.h`.

### 5.518.2.4 `typedef _Ios_Openmode std::ios_base::openmode`

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 363 of file `ios_base.h`.

### 5.518.2.5 `typedef _Ios_Seekdir std::ios_base::seekdir`

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 395 of file `ios_base.h`.

### 5.518.3 Member Enumeration Documentation

#### 5.518.3.1 `enum std::ios_base::event`

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 421 of file `ios_base.h`.

### 5.518.4 Constructor & Destructor Documentation

#### 5.518.4.1 `virtual std::ios_base::~ios_base ( ) [virtual]`

Invokes each callback with `erase_event`. Destroys local storage.

Note that the `ios_base` object for the standard streams never gets destroyed. As a result, any callbacks registered with the standard streams will not get invoked with `erase_event` (unless `copyfmt` is used).

### 5.518.5 Member Function Documentation

#### 5.518.5.1 `const locale& std::ios_base::_M_getloc ( ) const [inline]`

Locale access.

#### Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 708 of file `ios_base.h`.

Referenced by `std::money_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get<`

\_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::time\_put< \_CharT, \_OutIter >::do\_put(), std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::time\_put< \_CharT, \_OutIter >::put().

#### 5.518.5.2 fmtflags std::ios\_base::flags ( ) const [inline]

Access to format flags.

##### Returns

The format control flags for both input and output.

Definition at line 553 of file ios\_base.h.

Referenced by std::basic\_ios< \_CharT, \_Traits >::copyfmt(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::operator<<(), std::operator>>(), and std::basic\_istream< \_CharT, \_Traits >::sentry::sentry().

#### 5.518.5.3 fmtflags std::ios\_base::flags ( fmtflags \_\_fmtfl ) [inline]

Setting new format flags all at once.

##### Parameters

*fmtfl* The new flags to set.

##### Returns

The previous format control flags.

This function overwrites all the format flags with *fmtfl*.

Definition at line 564 of file ios\_base.h.

#### 5.518.5.4 locale std::ios\_base::getloc ( ) const [inline]

Locale access.

##### Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 697 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::money_put<_CharT, _OutIter>::do_put()`, `std::basic_ios<_CharT, _Traits>::imbue()`, `std::operator>>()`, and `std::ws()`.

#### 5.518.5.5 locale std::ios\_base::imbue ( const locale & \_\_loc ) throw ()

Setting a new locale.

##### Parameters

*loc* The new locale.

##### Returns

The previous locale.

Sets the new locale for this stream, and then invokes each callback with `imbue_event`.

Reimplemented in [std::basic\\_ios<\\_CharT, \\_Traits>](#), and [std::basic\\_ios<char, \\_Traits>](#).

#### 5.518.5.6 long& std::ios\_base::iword ( int \_\_ix ) [inline]

Access to integer array.

##### Parameters

*\_\_ix* Index into the array.

##### Returns

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 743 of file `ios_base.h`.

**5.518.5.7 streamsize std::ios\_base::precision ( ) const [inline]**

Flags access.

**Returns**

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 623 of file ios\_base.h.

Referenced by std::basic\_ios< \_CharT, \_Traits >::copyfmt(), and std::operator<<().

**5.518.5.8 streamsize std::ios\_base::precision ( streamsize \_\_prec ) [inline]**

Changing flags.

**Parameters**

*prec* The new precision value.

**Returns**

The previous value of [precision\(\)](#).

Definition at line 632 of file ios\_base.h.

**5.518.5.9 void\*& std::ios\_base::pword ( int \_\_ix ) [inline]**

Access to void pointer array.

**Parameters**

*\_\_ix* Index into the array.

**Returns**

A reference to a void\* associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 764 of file `ios_base.h`.

#### 5.518.5.10 void std::ios\_base::register\_callback ( event\_callback \_\_fn, int \_\_index )

Add the callback `__fn` with parameter `__index`.

##### Parameters

`__fn` The function to add.

`__index` The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

#### 5.518.5.11 fmtflags std::ios\_base::setf ( fmtflags \_\_fmtfl, fmtflags \_\_mask ) [inline]

Setting new format flags.

##### Parameters

`__fmtfl` Additional flags to set.

`__mask` The flags mask for `__fmtfl`.

##### Returns

The previous format control flags.

This function clears `__mask` in the format flags, then sets `__fmtfl & __mask`. An example mask is `ios_base::adjustfield`.

Definition at line 597 of file `ios_base.h`.



**5.518.5.12 fmtflags std::ios\_base::setf ( fmtflags \_\_fmtfl ) [inline]**

Setting new format flags.

**Parameters**

*fmtfl* Additional flags to set.

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 580 of file ios\_base.h.

Referenced by std::dec(), std::fixed(), std::hex(), std::left(), std::oct(), std::right(), std::scientific(), std::showbase(), std::showpoint(), std::showpos(), std::skipws(), std::unitbuf(), and std::uppercase().

**5.518.5.13 static bool std::ios\_base::sync\_with\_stdio ( bool \_\_sync = true ) [static]**

Interaction with the standard C I/O objects.

**Parameters**

*sync* Whether to synchronize or not.

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

**5.518.5.14 void std::ios\_base::unsetf ( fmtflags \_\_mask ) [inline]**

Clearing format flags.

**Parameters**

*mask* The flags to unset.

This function clears *mask* in the format flags.

Definition at line 612 of file ios\_base.h.

Referenced by std::noboolalpha(), std::noshowbase(), std::noshowpoint(), std::noshowpos(), std::noskipws(), std::nounitbuf(), and std::nouppercase().

**5.518.5.15 streamsize std::ios\_base::width ( ) const [inline]**

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 646 of file ios\_base.h.

Referenced by std::basic\_ios< \_CharT, \_Traits >::copyfmt(), std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::operator>>().

**5.518.5.16 streamsize std::ios\_base::width ( streamsize \_\_wide ) [inline]**

Changing flags.

**Parameters**

*wide* The new width value.

**Returns**

The previous value of [width\(\)](#).

Definition at line 655 of file ios\_base.h.

**5.518.5.17 static int std::ios\_base::xalloc ( ) throw () [static]**

Access to unique indices.

### Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

### 5.518.6 Member Data Documentation

#### 5.518.6.1 const fmtflags std::ios\_base::adjustfield [static]

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Definition at line 312 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

#### 5.518.6.2 const openmode std::ios\_base::app [static]

Seek to end before each write.

Definition at line 366 of file `ios_base.h`.

#### 5.518.6.3 const openmode std::ios\_base::ate [static]

Open and seek to end immediately after opening.

Definition at line 369 of file `ios_base.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::open()`.

#### 5.518.6.4 const iostate std::ios\_base::badbit [static]

Indicates a loss of integrity in an input or output sequence (such /// as an irrecoverable read error from a file).

Definition at line 336 of file ios\_base.h.

Referenced by std::basic\_ostream< char >::\_M\_write(), std::basic\_ios< char, \_Traits >::bad(), std::basic\_ios< \_CharT, \_Traits >::clear(), std::basic\_ios< char, \_Traits >::fail(), std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_ios< \_CharT, \_Traits >::init(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::operator<<(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::sync(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_ostream< \_CharT, \_Traits >::tellp(), std::basic\_istream< \_CharT, \_Traits >::unget(), std::basic\_ostream< \_CharT, \_Traits >::write(), and std::basic\_ostream< \_CharT, \_Traits >::sentry::~sentry().

#### 5.518.6.5 const fmtflags std::ios\_base::basefield [static]

A mask of dec|oct|hex. Useful for the 2-arg form of setf.

Definition at line 315 of file ios\_base.h.

Referenced by std::dec(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::hex(), std::oct(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

#### 5.518.6.6 const seekdir std::ios\_base::beg [static]

Request a seek relative to the beginning of the stream.

Definition at line 398 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::seekpos().

#### 5.518.6.7 const openmode std::ios\_base::binary [static]

Perform input and output in binary mode (as opposed to text mode). /// This is probably not what you think it is; see ///

<http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 374 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::showmanyc().

#### 5.518.6.8 const fmtflags std::ios\_base::boolalpha [static]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 260 of file ios\_base.h.

Referenced by std::boolalpha(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::noboolalpha().

#### 5.518.6.9 const seekdir std::ios\_base::cur [static]

Request a seek relative to the current position within the sequence.

Definition at line 401 of file ios\_base.h.

Referenced by std::basic\_filebuf< \_CharT, \_Traits >::imbue(), std::basic\_filebuf< \_CharT, \_Traits >::overflow(), std::basic\_filebuf< \_CharT, \_Traits >::pbackfail(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_filebuf< \_CharT, \_Traits >::seekoff(), std::basic\_istream< \_CharT, \_Traits >::tellg(), and std::basic\_ostream< \_CharT, \_Traits >::tellp().

#### 5.518.6.10 const fmtflags std::ios\_base::dec [static]

Converts integer input or generates integer output in decimal base.

Definition at line 263 of file ios\_base.h.

Referenced by std::dec().

#### 5.518.6.11 const seekdir std::ios\_base::end [static]

Request a seek relative to the current end of the sequence.

Definition at line 404 of file ios\_base.h.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`.

#### 5.518.6.12 `const iostate std::ios_base::eofbit` [static]

Indicates that an input operation reached the end of an input sequence.

Definition at line 339 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_time()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_ios< char, _Traits >::eof()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

#### 5.518.6.13 `const iostate std::ios_base::failbit` [static]

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 344 of file `ios_base.h`.

Referenced by `std::basic_fstream< _CharT, _Traits >::close()`, `std::basic_ofstream< _CharT, _Traits >::close()`, `std::basic_ifstream< _CharT, _Traits >::close()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_time()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_ios< char, _Traits >::fail()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_fstream< _CharT, _Traits >::open()`, `std::basic_ofstream< _CharT, _Traits >::open()`, `std::basic_ifstream< _CharT, _Traits >::open()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

**5.518.6.14 const fmtflags std::ios\_base::fixed [static]**

Generate floating-point output in fixed-point notation.

Definition at line 266 of file ios\_base.h.

Referenced by std::fixed().

**5.518.6.15 const fmtflags std::ios\_base::floatfield [static]**

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 318 of file ios\_base.h.

Referenced by std::fixed(), and std::scientific().

**5.518.6.16 const iostate std::ios\_base::goodbit [static]**

Indicates all is well.

Definition at line 347 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_ios< \_CharT, \_Traits >::init(), std::basic\_ostream< \_CharT, \_Traits >::operator<<(), std::operator>>(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::basic\_istream< \_CharT, \_Traits >::peek(), std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::read(), std::basic\_istream< \_CharT, \_Traits >::readsome(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_ostream< \_CharT, \_Traits >::seekg(), std::basic\_istream< \_CharT, \_Traits >::seekp(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< \_CharT, \_Traits >::sync(), and std::basic\_istream< \_CharT, \_Traits >::unget().

**5.518.6.17 const fmtflags std::ios\_base::hex [static]**

Converts integer input or generates integer output in hexadecimal base.

Definition at line 269 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::hex()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

#### 5.518.6.18 `const openmode std::ios_base::in [static]`

Open for input. Default for `ifstream` and `fstream`.

Definition at line 377 of file `ios_base.h`.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_set_buffer()`, `std::basic_ifstream< _CharT, _Traits >::open()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

#### 5.518.6.19 `const fmtflags std::ios_base::internal [static]`

Adds fill characters at a designated internal point in certain `///` generated output, or identical to `right` if no such point is `///` designated.

Definition at line 274 of file `ios_base.h`.

Referenced by `std::internal()`.

#### 5.518.6.20 `const fmtflags std::ios_base::left [static]`

Adds fill characters on the right (final positions) of certain `///` generated output. (I.e., the thing you print is flush left.).

Definition at line 278 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, and `std::left()`.



**5.518.6.21 const fmtflags std::ios\_base::oct [static]**

Converts integer input or generates integer output in octal base.

Definition at line 281 of file ios\_base.h.

Referenced by std::oct(), and std::basic\_ostream< \_CharT, \_Traits >::operator<<().

**5.518.6.22 const openmode std::ios\_base::out [static]**

Open for output. Default for ofstream and fstream.

Definition at line 380 of file ios\_base.h.

Referenced by std::basic\_filebuf< char\_type, traits\_type >::\_M\_set\_buffer(), std::basic\_ofstream< \_CharT, \_Traits >::open(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::overflow(), std::basic\_filebuf< \_CharT, \_Traits >::overflow(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::pbackfail(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos(), std::basic\_ostream< \_CharT, \_Traits >::tellp(), and std::basic\_filebuf< \_CharT, \_Traits >::xsputn().

**5.518.6.23 const fmtflags std::ios\_base::right [static]**

Adds fill characters on the left (initial positions) of certain /// generated output. (I.e., the thing you print is flush right.).

Definition at line 285 of file ios\_base.h.

Referenced by std::right().

**5.518.6.24 const fmtflags std::ios\_base::scientific [static]**

Generates floating-point output in scientific notation.

Definition at line 288 of file ios\_base.h.

Referenced by std::scientific().

**5.518.6.25 `const fmtflags std::ios_base::showbase` [static]**

Generates a prefix indicating the numeric base of generated integer /// output.

Definition at line 292 of file `ios_base.h`.

Referenced by `std::noshowbase()`, and `std::showbase()`.

**5.518.6.26 `const fmtflags std::ios_base::showpoint` [static]**

Generates a decimal-point character unconditionally in generated /// floating-point output.

Definition at line 296 of file `ios_base.h`.

Referenced by `std::noshowpoint()`, and `std::showpoint()`.

**5.518.6.27 `const fmtflags std::ios_base::showpos` [static]**

Generates a + sign in non-negative generated numeric output.

Definition at line 299 of file `ios_base.h`.

Referenced by `std::noshowpos()`, and `std::showpos()`.

**5.518.6.28 `const fmtflags std::ios_base::skipws` [static]**

Skips leading white space before certain input operations.

Definition at line 302 of file `ios_base.h`.

Referenced by `std::noskipws()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, and `std::skipws()`.

**5.518.6.29 `const openmode std::ios_base::trunc` [static]**

Open for input. Default for `ofstream`.

Definition at line 383 of file `ios_base.h`.

**5.518.6.30 const fmtflags std::ios\_base::unitbuf [static]**

Flushes output after each output operation.

Definition at line 305 of file ios\_base.h.

Referenced by std::nunitbuf(), std::unitbuf(), and std::basic\_ostream< \_CharT, \_Traits >::sentry::~sentry().

**5.518.6.31 const fmtflags std::ios\_base::uppercase [static]**

Replaces certain lowercase letters with their uppercase equivalents /// in generated output.

Definition at line 309 of file ios\_base.h.

Referenced by std::num\_put< \_CharT, \_OutIter >::do\_put(), std::nouppercase(), and std::uppercase().

The documentation for this class was generated from the following file:

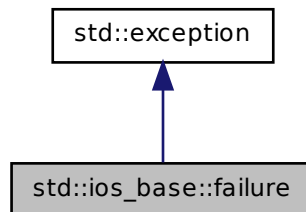
- [ios\\_base.h](#)

**5.519 std::ios\_base::failure Class Reference**

These are thrown to indicate problems with io.

27.4.2.1.1 Class [ios\\_base::failure](#).

Inheritance diagram for std::ios\_base::failure:



### Public Member Functions

- **failure** (const [string](#) &\_\_str) throw ()
- virtual const char \* **what** () const throw ()

#### 5.519.1 Detailed Description

These are thrown to indicate problems with io.

27.4.2.1.1 Class [ios\\_base::failure](#).

Definition at line 211 of file [ios\\_base.h](#).

#### 5.519.2 Member Function Documentation

##### 5.519.2.1 virtual const char\* `std::ios_base::failure::what ( ) const throw ()` [[virtual](#)]

Returns a C-style character string describing the general cause of the current error.

Reimplemented from [std::exception](#).

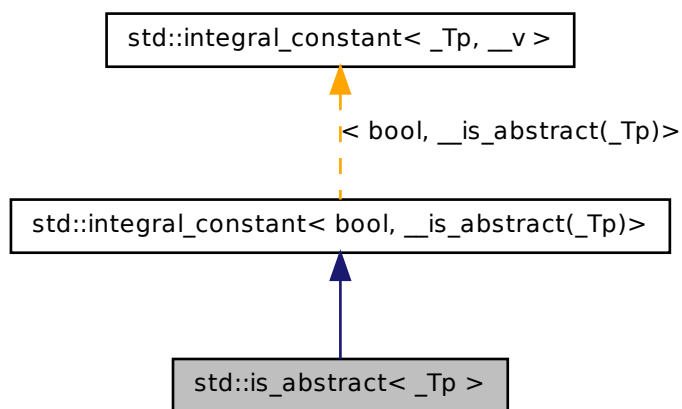
The documentation for this class was generated from the following file:

- [ios\\_base.h](#)

### 5.520 `std::is_abstract<_Tp>` Struct Template Reference

[is\\_abstract](#)

Inheritance diagram for std::is\_abstract< \_Tp >:



### Public Types

- typedef `integral_constant< bool, __v >` **type**
- typedef `bool` **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr `bool` **value**

#### 5.520.1 Detailed Description

**template<typename \_Tp> struct std::is\_abstract< \_Tp >**

[`is\_abstract`](#)

Definition at line 355 of file `type_traits`.

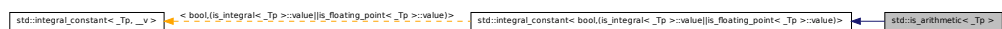
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.521 `std::is_arithmetic< _Tp >` Struct Template Reference

[is\\_arithmetic](#)

Inheritance diagram for `std::is_arithmetic< _Tp >`:



### Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr bool **value**

#### 5.521.1 Detailed Description

**template<typename \_Tp> struct `std::is_arithmetic< _Tp >`**

[is\\_arithmetic](#)

Definition at line 271 of file `type_traits`.

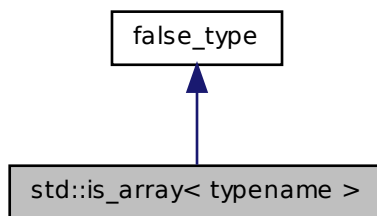
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.522 `std::is_array< typename >` Struct Template Reference

[is\\_array](#)

Inheritance diagram for `std::is_array< typename >`:



### Public Types

- typedef [integral\\_constant](#)< `_Tp`, `__v` > **type**
- typedef `_Tp` **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr `_Tp` **value**

#### 5.522.1 Detailed Description

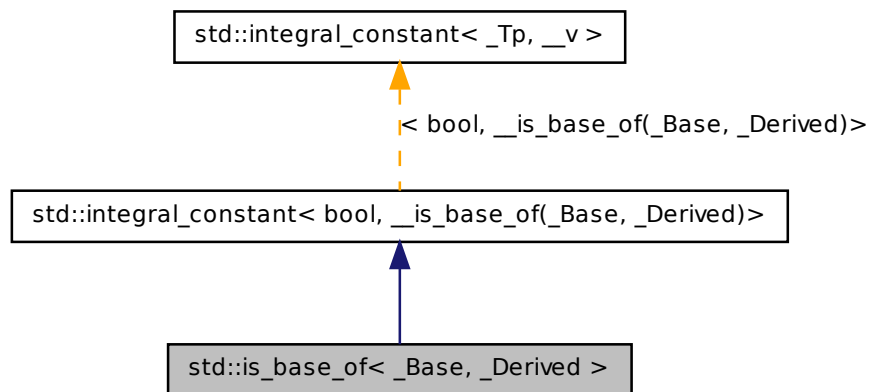
**template<typename> struct `std::is_array< typename >`**

[is\\_array](#)

Definition at line 151 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

**5.523 `std::is_base_of< _Base, _Derived >` Struct Template Reference**[is\\_base\\_of](#)Inheritance diagram for `std::is_base_of< _Base, _Derived >`:**Public Types**

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

**Public Member Functions**

- constexpr **operator value\_type** ()

**Static Public Attributes**

- static constexpr bool **value**



### 5.523.1 Detailed Description

```
template<typename _Base, typename _Derived> struct std::is_base_of< _Base,
_Derived >
```

[is\\_base\\_of](#)

Definition at line 757 of file type\_traits.

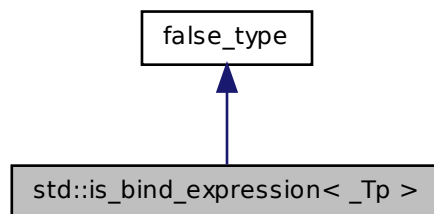
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.524 std::is\_bind\_expression< \_Tp > Struct Template Reference

Determines if the given type `_Tp` is a function object should be treated as a subexpression when evaluating calls to function objects returned by [bind\(\)](#). [TR1 3.6.1].

Inheritance diagram for `std::is_bind_expression< _Tp >`:



### Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr `_Tp` `value`

#### 5.524.1 Detailed Description

`template<typename _Tp> struct std::is_bind_expression< _Tp >`

Determines if the given type `_Tp` is a function object should be treated as a subexpression when evaluating calls to function objects returned by `bind()`. [TR1 3.6.1].

Definition at line 824 of file `functional`.

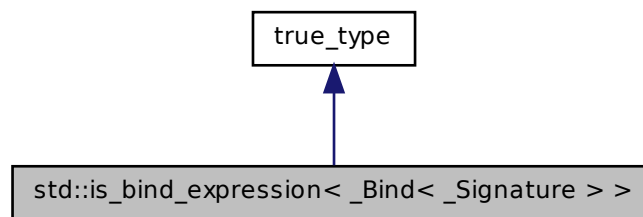
The documentation for this struct was generated from the following file:

- `functional`

### 5.525 `std::is_bind_expression< _Bind< _Signature > >` Struct Template Reference

Class template `_Bind` is always a bind expression.

Inheritance diagram for `std::is_bind_expression< _Bind< _Signature > >`:



### Public Types

- typedef `integral_constant< _Tp, __v >` `type`
- typedef `_Tp` `value_type`

### Public Member Functions

- `constexpr operator value_type ()`

### Static Public Attributes

- `static constexpr _Tp value`

#### 5.525.1 Detailed Description

`template<typename _Signature> struct std::is_bind_expression< _Bind< _Signature > >`

Class template `_Bind` is always a bind expression.

Definition at line 1408 of file `functional`.

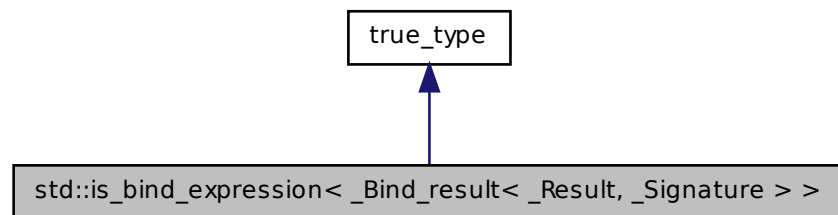
The documentation for this struct was generated from the following file:

- [functional](#)

### 5.526 `std::is_bind_expression< _Bind_result< _Result, _Signature > > Struct` Template Reference

Class template `_Bind` is always a bind expression.

Inheritance diagram for `std::is_bind_expression< _Bind_result< _Result, _Signature > >`:



**Public Types**

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

**Public Member Functions**

- constexpr **operator value\_type** ()

**Static Public Attributes**

- static constexpr \_Tp **value**

**5.526.1 Detailed Description**

**template<typename \_Result, typename \_Signature> struct std::is\_bind\_expression< \_Bind\_result< \_Result, \_Signature > >**

Class template `_Bind` is always a bind expression.

Definition at line 1416 of file `functional`.

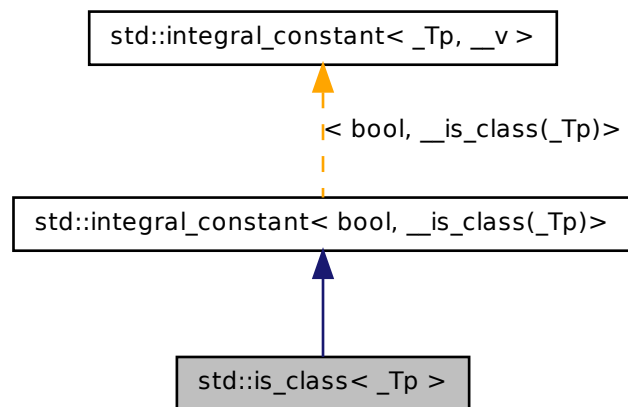
The documentation for this struct was generated from the following file:

- [functional](#)

**5.527 `std::is_class< _Tp >` Struct Template Reference**

[is\\_class](#)

Inheritance diagram for std::is\_class< \_Tp >:



### Public Types

- typedef `integral_constant< bool, __v >` **type**
- typedef `bool` **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr `bool` **value**

#### 5.527.1 Detailed Description

**template<typename \_Tp> struct std::is\_class< \_Tp >**

[`is\_class`](#)

Definition at line 222 of file `type_traits`.

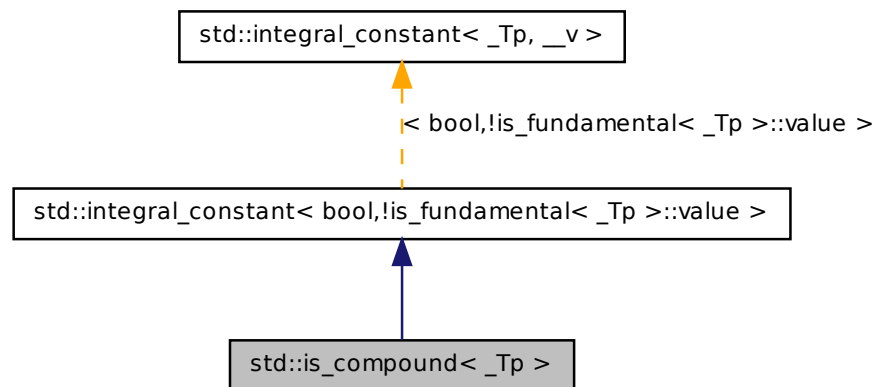
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.528 `std::is_compound< _Tp >` Struct Template Reference

### [is\\_compound](#)

Inheritance diagram for `std::is_compound< _Tp >`:



### Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr bool **value**

### 5.528.1 Detailed Description

`template<typename _Tp> struct std::is_compound< _Tp >`

[is\\_compound](#)

Definition at line 307 of file `type_traits`.

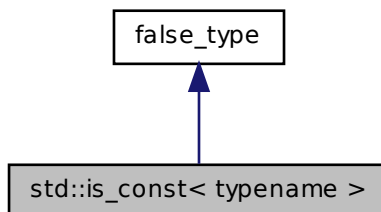
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.529 `std::is_const< typename >` Struct Template Reference

[is\\_const](#)

Inheritance diagram for `std::is_const< typename >`:



### Public Types

- typedef [integral\\_constant< \\_Tp, \\_\\_v >](#) **type**
- typedef `_Tp` **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr `_Tp` **value**

## 5.529.1 Detailed Description

```
template<typename> struct std::is_const< typename >
```

[is\\_const](#)

Definition at line 325 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

5.530 `std::is_constructible< _Tp, _Args >` Struct Template Reference

[is\\_constructible](#)

Inheritance diagram for `std::is_constructible< _Tp, _Args >`:



## Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

## Public Member Functions

- constexpr **operator value\_type** ()

## Static Public Attributes

- static constexpr bool **value**

## 5.530.1 Detailed Description

```
template<typename _Tp, typename... _Args> struct std::is_constructible< _Tp,
_Args >
```

[is\\_constructible](#)

Definition at line 683 of file `type_traits`.



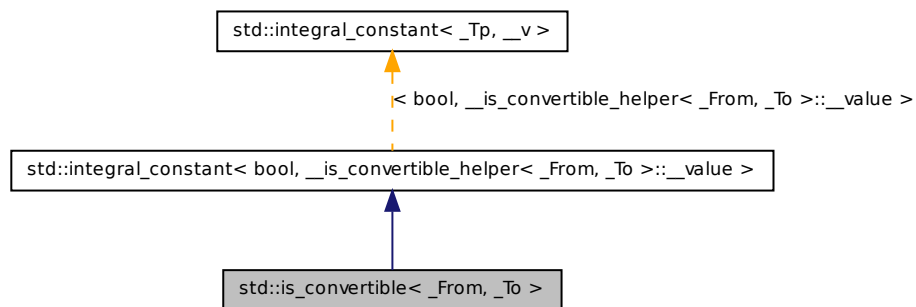
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.531 `std::is_convertible< _From, _To >` Struct Template Reference

[is\\_convertible](#)

Inheritance diagram for `std::is_convertible< _From, _To >`:



### Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr bool **value**

### 5.531.1 Detailed Description

```
template<typename _From, typename _To> struct std::is_convertible< _From,
_To >
```

[is\\_convertible](#)

Definition at line 789 of file type\_traits.

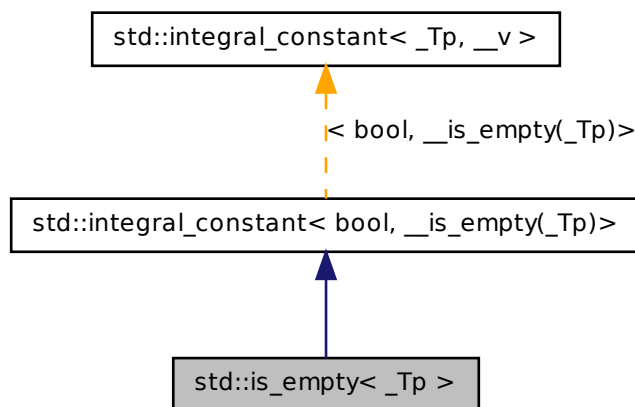
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.532 std::is\_empty< \_Tp > Struct Template Reference

[is\\_empty](#)

Inheritance diagram for std::is\_empty< \_Tp >:



### Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

**Public Member Functions**

- `constexpr operator value_type ()`

**Static Public Attributes**

- `static constexpr bool value`

**5.532.1 Detailed Description**

`template<typename _Tp> struct std::is_empty< _Tp >`

[is\\_empty](#)

Definition at line 343 of file `type_traits`.

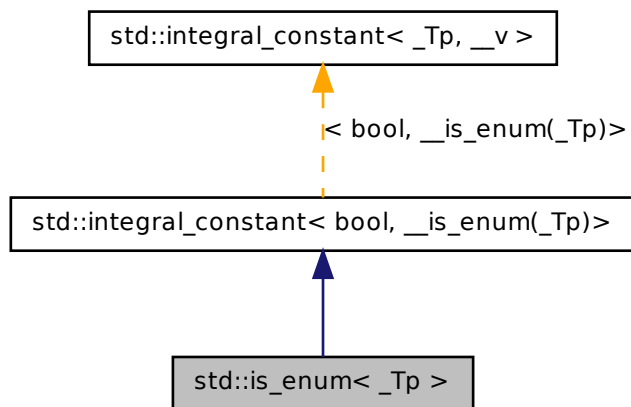
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

**5.533 `std::is_enum< _Tp >` Struct Template Reference**

[is\\_enum](#)

Inheritance diagram for std::is\_enum< \_Tp >:



### Public Types

- typedef `integral_constant< bool, __v >` **type**
- typedef `bool` **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr `bool` **value**

#### 5.533.1 Detailed Description

**template<typename \_Tp> struct std::is\_enum< \_Tp >**

`is_enum`

Definition at line 210 of file `type_traits`.

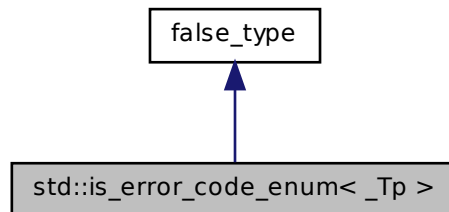
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.534 `std::is_error_code_enum< _Tp >` Struct Template Reference

[is\\_error\\_code\\_enum](#)

Inheritance diagram for `std::is_error_code_enum< _Tp >`:



### Public Types

- typedef [integral\\_constant](#)< `_Tp`, `__v` > **type**
- typedef `_Tp` **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr `_Tp` **value**

## 5.535 `std::is_error_code_enum< future_errc >` Struct Template Reference 2779

### 5.534.1 Detailed Description

`template<typename _Tp> struct std::is_error_code_enum< _Tp >`

[is\\_error\\_code\\_enum](#)

Definition at line 54 of file `system_error`.

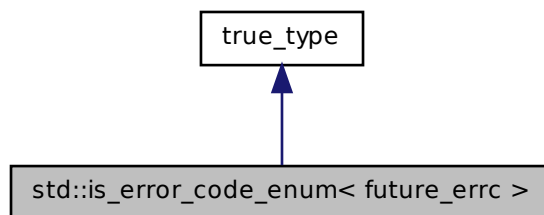
The documentation for this struct was generated from the following file:

- [system\\_error](#)

## 5.535 `std::is_error_code_enum< future_errc >` Struct Template Reference

Specialization.

Inheritance diagram for `std::is_error_code_enum< future_errc >`:



### Public Types

- typedef [integral\\_constant](#)< `_Tp`, `__v` > `type`
- typedef `_Tp` `value_type`

### Public Member Functions

- constexpr `operator value_type` ()

## Static Public Attributes

- static constexpr `_Tp` `value`

### 5.535.1 Detailed Description

`template<> struct std::is_error_code_enum< future_errc >`

Specialization.

Definition at line 71 of file `future`.

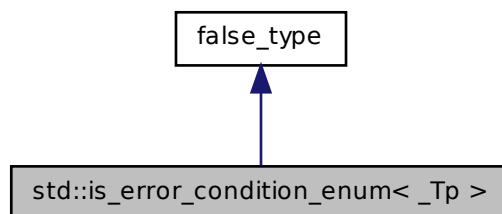
The documentation for this struct was generated from the following file:

- [future](#)

## 5.536 `std::is_error_condition_enum< _Tp >` Struct Template Reference

[is\\_error\\_condition\\_enum](#)

Inheritance diagram for `std::is_error_condition_enum< _Tp >`:



## Public Types

- typedef [integral\\_constant](#)< `_Tp`, `__v` > `type`
- typedef `_Tp` `value_type`

## 5.537 `std::is_explicitly_convertible< _From, _To >` Struct Template Reference

### Public Member Functions

- `constexpr operator value_type ()`

### Static Public Attributes

- `static constexpr _Tp value`

#### 5.536.1 Detailed Description

`template<typename _Tp> struct std::is_error_condition_enum< _Tp >`

[is\\_error\\_condition\\_enum](#)

Definition at line 58 of file `system_error`.

The documentation for this struct was generated from the following file:

- [system\\_error](#)

## 5.537 `std::is_explicitly_convertible< _From, _To >` Struct Template Reference

[is\\_explicitly\\_convertible](#)

Inheritance diagram for `std::is_explicitly_convertible< _From, _To >`:



### Public Types

- `typedef integral\_constant< bool, __v > type`
- `typedef bool value_type`

### Public Member Functions

- `constexpr operator value_type ()`

### Static Public Attributes

- `static constexpr bool value`



### 5.537.1 Detailed Description

```
template<typename _From, typename _To> struct std::is_explicitly_
convertible< _From, _To >
```

[is\\_explicitly\\_convertible](#)

Definition at line 796 of file `type_traits`.

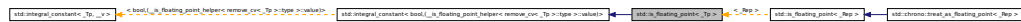
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.538 `std::is_floating_point< _Tp >` Struct Template Reference

[is\\_floating\\_point](#)

Inheritance diagram for `std::is_floating_point< _Tp >`:



### Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr bool **value**

### 5.538.1 Detailed Description

```
template<typename _Tp> struct std::is_floating_point< _Tp >
```

[is\\_floating\\_point](#)

Definition at line 144 of file `type_traits`.

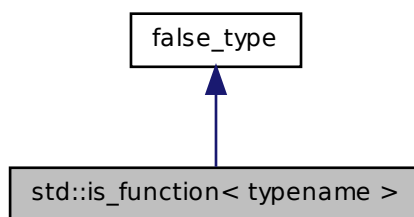
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.539 `std::is_function< typename >` Struct Template Reference

### [is\\_function](#)

Inheritance diagram for `std::is_function< typename >`:



### Public Types

- typedef [integral\\_constant](#)< `_Tp`, `__v` > **type**
- typedef `_Tp` **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr `_Tp` **value**

#### 5.539.1 Detailed Description

`template<typename> struct std::is_function< typename >`

### [is\\_function](#)

Definition at line 228 of file `type_traits`.

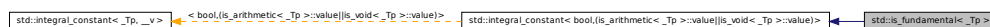
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.540 `std::is_fundamental<_Tp>` Struct Template Reference

[is\\_fundamental](#)

Inheritance diagram for `std::is_fundamental<_Tp>`:



### Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr bool **value**

#### 5.540.1 Detailed Description

`template<typename _Tp> struct std::is_fundamental<_Tp>`

[is\\_fundamental](#)

Definition at line 278 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.541 `std::is_integral< _Tp >` Struct Template Reference

[is\\_integral](#)

Inheritance diagram for `std::is_integral< _Tp >`:



### Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr bool **value**

#### 5.541.1 Detailed Description

`template<typename _Tp> struct std::is_integral< _Tp >`

[is\\_integral](#)

Definition at line 130 of file `type_traits`.

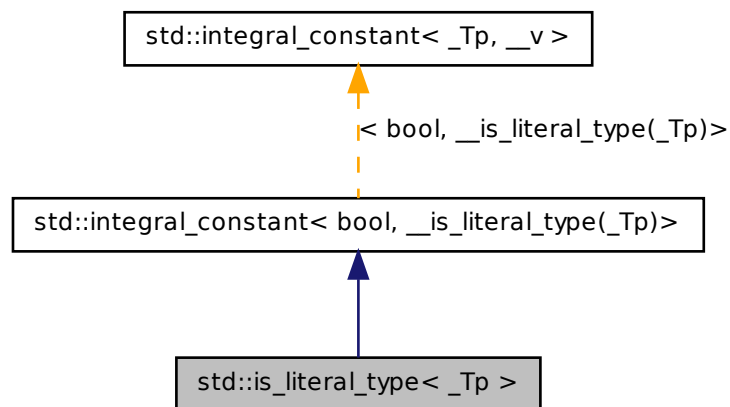
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.542 `std::is_literal_type< _Tp >` Struct Template Reference

[is\\_literal\\_type](#)

Inheritance diagram for `std::is_literal_type< _Tp >`:



### Public Types

- typedef `integral_constant< bool, __v >` **type**
- typedef `bool` **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr `bool` **value**

#### 5.542.1 Detailed Description

`template<typename _Tp> struct std::is_literal_type< _Tp >`

[`is\_literal\_type`](#)

Definition at line 643 of file `type_traits`.

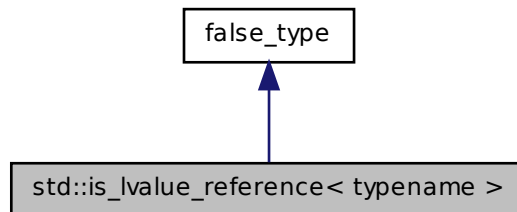
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.543 `std::is_lvalue_reference< typename >` Struct Template Reference

[is\\_lvalue\\_reference](#)

Inheritance diagram for `std::is_lvalue_reference< typename >`:



### Public Types

- typedef [integral\\_constant](#)< `_Tp`, `__v` > **type**
- typedef `_Tp` **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr `_Tp` **value**

## 5.544 `std::is_member_function_pointer< _Tp >` Struct Template Reference 2788

### 5.543.1 Detailed Description

`template<typename> struct std::is_lvalue_reference< typename >`

[is\\_lvalue\\_reference](#)

Definition at line 515 of file `type_traits`.

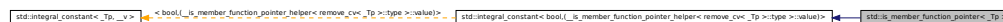
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.544 `std::is_member_function_pointer< _Tp >` Struct Template Reference

[is\\_member\\_function\\_pointer](#)

Inheritance diagram for `std::is_member_function_pointer< _Tp >`:



### Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr bool **value**

### 5.544.1 Detailed Description

`template<typename _Tp> struct std::is_member_function_pointer< _Tp >`

[is\\_member\\_function\\_pointer](#)

Definition at line 203 of file `type_traits`.

The documentation for this struct was generated from the following file:

## 5.545 `std::is_member_object_pointer< _Tp >` Struct Template Reference 2789

- [type\\_traits](#)

### 5.545 `std::is_member_object_pointer< _Tp >` Struct Template Reference

[is\\_member\\_object\\_pointer](#)

Inheritance diagram for `std::is_member_object_pointer< _Tp >`:



#### Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** ()

#### Static Public Attributes

- static constexpr bool **value**

#### 5.545.1 Detailed Description

`template<typename _Tp> struct std::is_member_object_pointer< _Tp >`

[is\\_member\\_object\\_pointer](#)

Definition at line 190 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 5.546 `std::is_nothrow_constructible< _Tp, _Args >` Struct Template Reference

[is\\_nothrow\\_constructible](#)



Inheritance diagram for `std::is_nothrow_constructible<_Tp, _Args>`:



### Public Types

- typedef `integral_constant<bool, __v>` **type**
- typedef `bool` **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr `bool` **value**

#### 5.546.1 Detailed Description

template<typename **\_Tp**, typename... **\_Args**> struct `std::is_nothrow_constructible<_Tp, _Args>`

`is_nothrow_constructible`

Definition at line 705 of file `type_traits`.

The documentation for this struct was generated from the following file:

- `type_traits`

### 5.547 `std::is_object<_Tp>` Struct Template Reference

`is_object`

Inheritance diagram for `std::is_object<_Tp>`:



**Public Types**

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

**Public Member Functions**

- constexpr **operator value\_type** ()

**Static Public Attributes**

- static constexpr bool **value**

**5.547.1 Detailed Description**

**template<typename \_Tp> struct std::is\_object<\_Tp>**

[is\\_object](#)

Definition at line 285 of file `type_traits`.

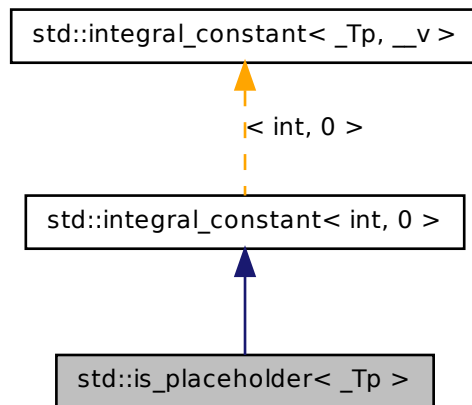
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

**5.548 `std::is_placeholder<_Tp>` Struct Template Reference**

Determines if the given type `_Tp` is a placeholder in a [bind\(\)](#) expression and, if so, which placeholder it is. [TR1 3.6.2].

Inheritance diagram for `std::is_placeholder<_Tp>`:



### Public Types

- typedef `integral_constant<int, __v>` **type**
- typedef `int` **value\_type**

### Public Member Functions

- `constexpr operator value_type ()`

### Static Public Attributes

- `static constexpr int` **value**

#### 5.548.1 Detailed Description

**template<typename \_Tp> struct `std::is_placeholder<_Tp>`**

Determines if the given type `_Tp` is a placeholder in a `bind()` expression and, if so, which placeholder it is. [TR1 3.6.2].

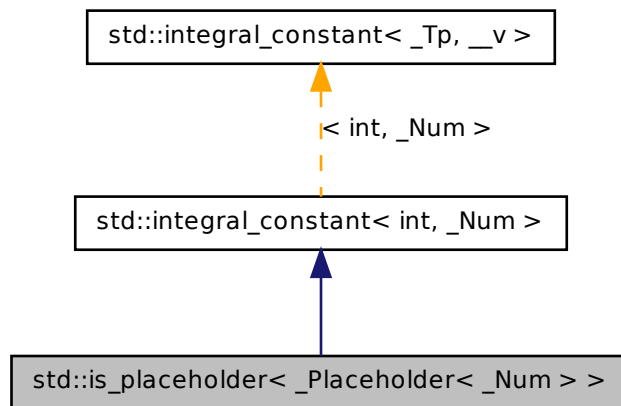
Definition at line 833 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

## 5.549 `std::is_placeholder<_Placeholder<_Num>>` Struct Template Reference

Inheritance diagram for `std::is_placeholder<_Placeholder<_Num>>`:



### Public Types

- typedef [integral\\_constant](#)< int, \_\_v > **type**
- typedef int **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr int **value**

## 5.549.1 Detailed Description

```
template<int _Num> struct std::is_placeholder< _Placeholder< _Num > >
```

Partial specialization of [is\\_placeholder](#) that provides the placeholder number for the placeholder objects defined by libstdc++.

Definition at line 893 of file functional.

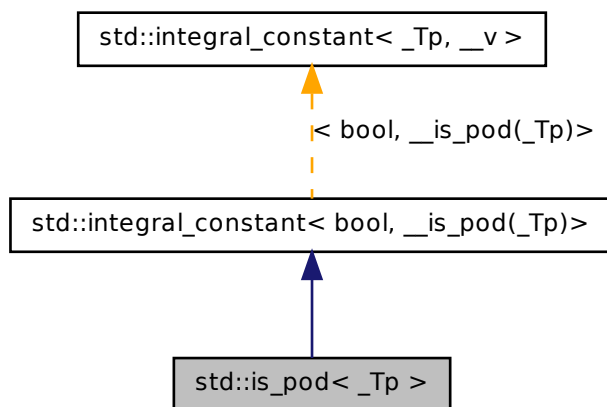
The documentation for this struct was generated from the following file:

- [functional](#)

## 5.550 std::is\_pod&lt; \_Tp &gt; Struct Template Reference

[is\\_pod](#)

Inheritance diagram for std::is\_pod< \_Tp >:



## Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

**Public Member Functions**

- constexpr **operator value\_type** ()

**Static Public Attributes**

- static constexpr bool **value**

**5.550.1 Detailed Description**

template<typename `_Tp`> struct `std::is_pod< _Tp >`

[is\\_pod](#)

Definition at line 637 of file `type_traits`.

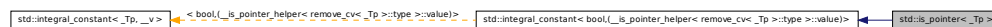
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

**5.551 `std::is_pointer< _Tp >` Struct Template Reference**

[is\\_pointer](#)

Inheritance diagram for `std::is_pointer< _Tp >`:

**Public Types**

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

**Public Member Functions**

- constexpr **operator value\_type** ()

**Static Public Attributes**

- static constexpr bool **value**

## 5.551.1 Detailed Description

```
template<typename _Tp> struct std::is_pointer< _Tp >
```

[is\\_pointer](#)

Definition at line 169 of file `type_traits`.

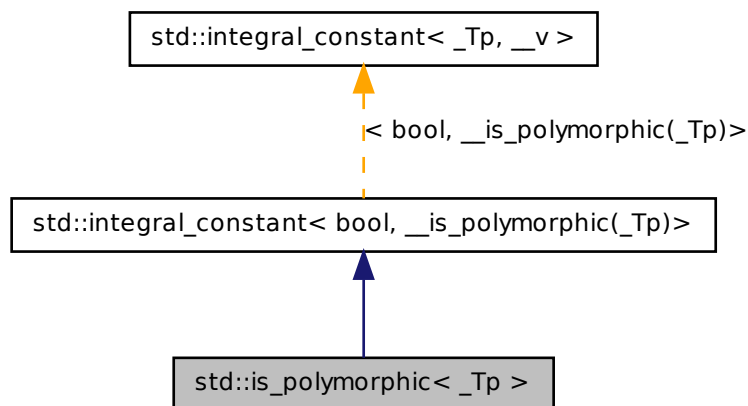
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

5.552 `std::is_polymorphic< _Tp >` Struct Template Reference

[is\\_polymorphic](#)

Inheritance diagram for `std::is_polymorphic< _Tp >`:



## Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

**Public Member Functions**

- constexpr **operator value\_type** ()

**Static Public Attributes**

- static constexpr bool **value**

**5.552.1 Detailed Description**

template<typename `_Tp`> struct `std::is_polymorphic< _Tp >`

[is\\_polymorphic](#)

Definition at line 349 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

**5.553 `std::is_reference< _Tp >` Struct Template Reference**

[is\\_reference](#)

Inheritance diagram for `std::is_reference< _Tp >`:

**Public Types**

- typedef [integral\\_constant](#)< bool, `__v` > **type**
- typedef bool **value\_type**

**Public Member Functions**

- constexpr **operator value\_type** ()

**Static Public Attributes**

- static constexpr bool **value**



### 5.553.1 Detailed Description

`template<typename _Tp> struct std::is_reference< _Tp >`

[is\\_reference](#)

Definition at line 535 of file `type_traits`.

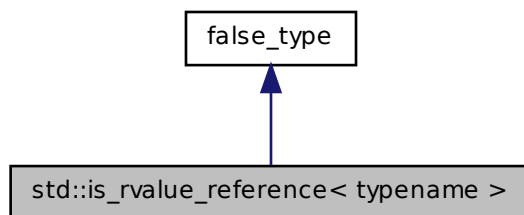
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.554 `std::is_rvalue_reference< typename >` Struct Template Reference

[is\\_rvalue\\_reference](#)

Inheritance diagram for `std::is_rvalue_reference< typename >`:



### Public Types

- typedef [integral\\_constant< \\_Tp, \\_\\_v >](#) **type**
- typedef `_Tp` **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

## Static Public Attributes

- static constexpr `_Tp` `value`

### 5.554.1 Detailed Description

`template<typename> struct std::is_rvalue_reference< typename >`

[is\\_rvalue\\_reference](#)

Definition at line 524 of file `type_traits`.

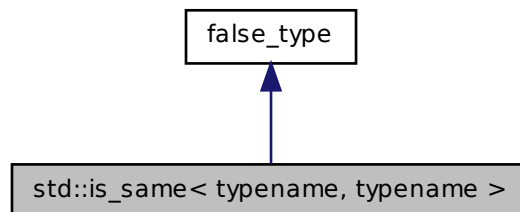
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.555 `std::is_same< typename, typename >` Struct Template Reference

[is\\_same](#)

Inheritance diagram for `std::is_same< typename, typename >`:



## Public Types

- typedef [integral\\_constant](#)< `_Tp`, `__v` > `type`
- typedef `_Tp` `value_type`

**Public Member Functions**

- constexpr **operator value\_type** ()

**Static Public Attributes**

- static constexpr **\_Tp value**

**5.555.1 Detailed Description**

template<typename, typename> struct std::is\_same< typename, typename >

[is\\_same](#)

Definition at line 406 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

**5.556 std::is\_scalar< \_Tp > Struct Template Reference**

[is\\_scalar](#)

Inheritance diagram for std::is\_scalar< \_Tp >:

**Public Types**

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

**Public Member Functions**

- constexpr **operator value\_type** ()

**Static Public Attributes**

- static constexpr bool **value**

**5.556.1 Detailed Description**

template<typename \_Tp> struct std::is\_scalar< \_Tp >

[is\\_scalar](#)

Definition at line 297 of file type\_traits.

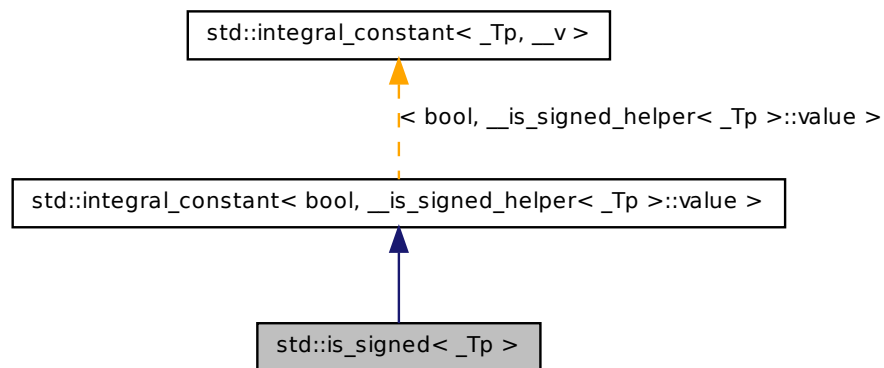
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

**5.557 std::is\_signed< \_Tp > Struct Template Reference**

[is\\_signed](#)

Inheritance diagram for std::is\_signed< \_Tp >:

**Public Types**

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

**Public Member Functions**

- constexpr **operator value\_type** ()

**Static Public Attributes**

- static constexpr bool **value**

**5.557.1 Detailed Description**

**template<typename \_Tp> struct std::is\_signed< \_Tp >**

[is\\_signed](#)

Definition at line 609 of file `type_traits`.

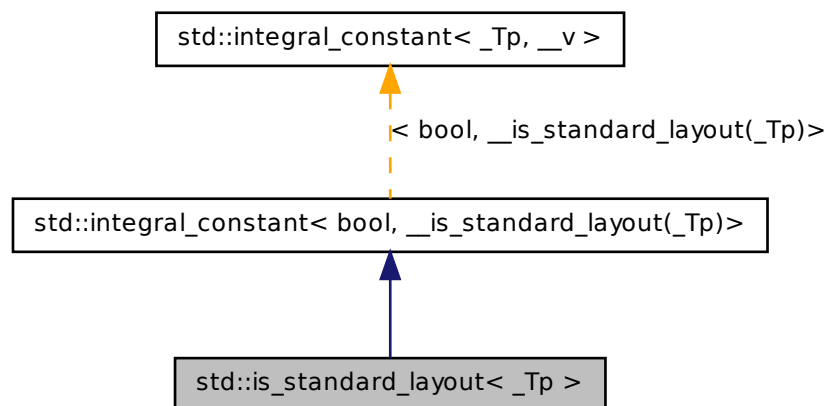
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

**5.558 `std::is_standard_layout< _Tp >` Struct Template Reference**

[is\\_standard\\_layout](#)

Inheritance diagram for `std::is_standard_layout< _Tp >`:



**Public Types**

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

**Public Member Functions**

- constexpr **operator value\_type** ()

**Static Public Attributes**

- static constexpr bool **value**

**5.558.1 Detailed Description**

**template<typename \_Tp> struct std::is\_standard\_layout<\_Tp>**

[is\\_standard\\_layout](#)

Definition at line 630 of file `type_traits`.

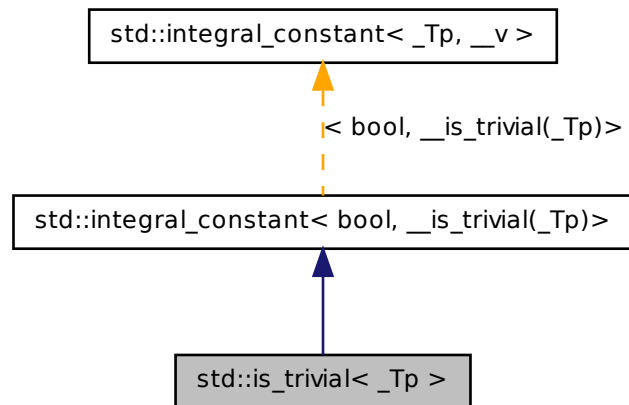
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

**5.559 `std::is_trivial<_Tp>` Struct Template Reference**

[is\\_trivial](#)

Inheritance diagram for std::is\_trivial< \_Tp >:



### Public Types

- typedef `integral_constant< bool, __v >` **type**
- typedef `bool` **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr `bool` **value**

#### 5.559.1 Detailed Description

**template<typename \_Tp> struct std::is\_trivial< \_Tp >**

[`is\_trivial`](#)

Definition at line 624 of file `type_traits`.

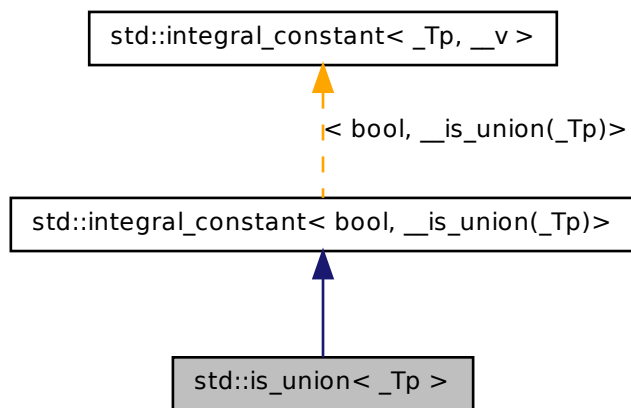
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.560 `std::is_union< _Tp >` Struct Template Reference

[is\\_union](#)

Inheritance diagram for `std::is_union< _Tp >`:



### Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr bool **value**



### 5.560.1 Detailed Description

**template<typename \_Tp> struct std::is\_union< \_Tp >**

[is\\_union](#)

Definition at line 216 of file type\_traits.

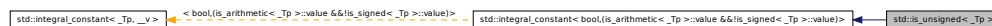
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.561 std::is\_unsigned< \_Tp > Struct Template Reference

[is\\_unsigned](#)

Inheritance diagram for std::is\_unsigned< \_Tp >:



### Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr bool **value**

### 5.561.1 Detailed Description

**template<typename \_Tp> struct std::is\_unsigned< \_Tp >**

[is\\_unsigned](#)

Definition at line 615 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.562 std::is\_void< \_Tp > Struct Template Reference

[is\\_void](#)

Inheritance diagram for std::is\_void< \_Tp >:



### Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr bool **value**

#### 5.562.1 Detailed Description

template<typename \_Tp> struct std::is\_void< \_Tp >

[is\\_void](#)

Definition at line 102 of file type\_traits.

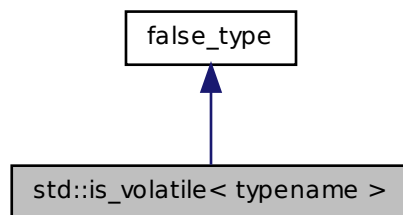
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.563 std::is\_volatile< typename > Struct Template Reference

[is\\_volatile](#)

Inheritance diagram for `std::is_volatile< typename >`:



### Public Types

- typedef [integral\\_constant](#)< `_Tp`, `__v` > `type`
- typedef `_Tp` `value_type`

### Public Member Functions

- constexpr `operator value_type` ()

### Static Public Attributes

- static constexpr `_Tp` `value`

#### 5.563.1 Detailed Description

`template<typename> struct std::is_volatile< typename >`

[is\\_volatile](#)

Definition at line 334 of file `type_traits`.

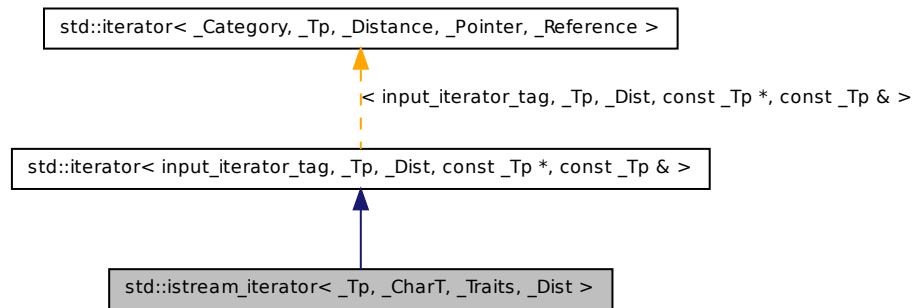
The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.564 `std::istream_iterator< _Tp, _CharT, _Traits, _Dist >` Class Template Reference

Provides input iterator semantics for streams.

Inheritance diagram for `std::istream_iterator< _Tp, _CharT, _Traits, _Dist >`:



### Public Types

- typedef `_CharT` **char\_type**
- typedef `_Dist` **difference\_type**
- typedef `basic_istream< _CharT, _Traits >` **istream\_type**
- typedef `input_iterator_tag` **iterator\_category**
- typedef `const _Tp *` **pointer**
- typedef `const _Tp &` **reference**
- typedef `_Traits` **traits\_type**
- typedef `_Tp` **value\_type**

### Public Member Functions

- constexpr `istream_iterator` ()
- `istream_iterator` (`istream_type` &\_\_s)
- **istream\_iterator** (const `istream_iterator` &\_\_obj)
- bool **\_M\_equal** (const `istream_iterator` &\_\_x) const
- const `_Tp` & **operator\*** () const
- `istream_iterator` & **operator++** ()
- `istream_iterator` **operator++** (int)
- const `_Tp *` **operator->** () const

### 5.564.1 Detailed Description

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename _Dist = ptrdiff_t> class std::istream_iterator< _Tp, _CharT, _Traits, _Dist >
```

Provides input iterator semantics for streams.

Definition at line 49 of file stream\_iterator.h.

### 5.564.2 Member Typedef Documentation

**5.564.2.1** `typedef _Dist std::iterator< input_iterator_tag , _Tp, _Dist , const _Tp * , const _Tp & >::difference_type [inherited]`

Distance between iterators is represented as this type.

Definition at line 126 of file stl\_iterator\_base\_types.h.

**5.564.2.2** `typedef input_iterator_tag std::iterator< input_iterator_tag , _Tp, _Dist , const _Tp * , const _Tp & >::iterator_category [inherited]`

One of the [tag types](#).

Definition at line 122 of file stl\_iterator\_base\_types.h.

**5.564.2.3** `typedef const _Tp * std::iterator< input_iterator_tag , _Tp, _Dist , const _Tp * , const _Tp & >::pointer [inherited]`

This type represents a pointer-to-value\_type.

Definition at line 128 of file stl\_iterator\_base\_types.h.

**5.564.2.4** `typedef const _Tp & std::iterator< input_iterator_tag , _Tp, _Dist , const _Tp * , const _Tp & >::reference [inherited]`

This type represents a reference-to-value\_type.

Definition at line 130 of file stl\_iterator\_base\_types.h.

## 5.565 std::istreambuf\_iterator< \_CharT, \_Traits > Class Template Reference 2811

**5.564.2.5** `typedef _Tp std::iterator< input_iterator_tag, _Tp, _Dist, const _Tp*, const _Tp & >::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 124 of file stl\_iterator\_base\_types.h.

### 5.564.3 Constructor & Destructor Documentation

**5.564.3.1** `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename _Dist = ptrdiff_t> constexpr std::istream_iterator< _Tp, _CharT, _Traits, _Dist >::istream_iterator( ) [inline]`

Construct end of input stream iterator.

Definition at line 64 of file stream\_iterator.h.

**5.564.3.2** `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename _Dist = ptrdiff_t> std::istream_iterator< _Tp, _CharT, _Traits, _Dist >::istream_iterator( istream_type & __s ) [inline]`

Construct start of input stream iterator.

Definition at line 68 of file stream\_iterator.h.

The documentation for this class was generated from the following file:

- [stream\\_iterator.h](#)

## 5.565 std::istreambuf\_iterator< \_CharT, \_Traits > Class Template Reference

Provides input iterator semantics for streambufs.

Inheritance diagram for std::istreambuf\_iterator< \_CharT, \_Traits >:



## Public Types

- typedef \_Traits::off\_type [difference\\_type](#)
- typedef [input\\_iterator\\_tag](#) iterator\_category
- typedef \_CharT \* [pointer](#)
- typedef \_CharT & [reference](#)
- typedef \_CharT [value\\_type](#)
  
- typedef \_CharT [char\\_type](#)
- typedef \_Traits [traits\\_type](#)
- typedef \_Traits::int\_type [int\\_type](#)
- typedef [basic\\_streambuf](#)< \_CharT, \_Traits > [streambuf\\_type](#)
- typedef [basic\\_istream](#)< \_CharT, \_Traits > [istream\\_type](#)

## Public Member Functions

- constexpr [istreambuf\\_iterator](#) () throw ()
- [istreambuf\\_iterator](#) ([istream\\_type](#) &\_\_s) throw ()
- [istreambuf\\_iterator](#) ([streambuf\\_type](#) \*\_\_s) throw ()
- bool [equal](#) (const [istreambuf\\_iterator](#) &\_\_b) const
- [char\\_type](#) operator\* () const
- [istreambuf\\_iterator](#) operator++ (int)
- [istreambuf\\_iterator](#) & operator++ ()

## Friends

- template<bool \_IsMove, typename \_CharT2 >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_char< \_CharT2 >::\_\_value, \_CharT2 \* >::\_\_type [\\_\\_copy\\_move\\_a2](#) ([istreambuf\\_iterator](#)< \_CharT2 >, [istreambuf\\_iterator](#)< \_CharT2 >, \_CharT2 \*)
- template<typename \_CharT2 >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_char< \_CharT2 >::\_\_value, [ostreambuf\\_iterator](#)< \_CharT2 > >::\_\_type [copy](#) ([istreambuf\\_iterator](#)< \_CharT2 >, [istreambuf\\_iterator](#)< \_CharT2 >, [ostreambuf\\_iterator](#)< \_CharT2 >)
- template<typename \_CharT2 >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_char< \_CharT2 >::\_\_value, [istreambuf\\_iterator](#)< \_CharT2 > >::\_\_type [find](#) ([istreambuf\\_iterator](#)< \_CharT2 >, [istreambuf\\_iterator](#)< \_CharT2 >, const \_CharT2 &)

### 5.565.1 Detailed Description

```
template<typename _CharT, typename _Traits> class std::istreambuf_iterator<
_CharT, _Traits >
```

Provides input iterator semantics for streambufs.

Definition at line 52 of file `streambuf_iterator.h`.

### 5.565.2 Member Typedef Documentation

**5.565.2.1** `template<typename _CharT, typename _Traits> typedef _CharT  
std::istreambuf_iterator<_CharT, _Traits>::char_type`

Public typedefs.

Definition at line 60 of file `streambuf_iterator.h`.

**5.565.2.2** `typedef _Traits::off_type std::iterator<input_iterator_tag, _CharT  
, _Traits::off_type, _CharT*, _CharT &>::difference_type  
[inherited]`

Distance between iterators is represented as this type.

Definition at line 126 of file `stl_iterator_base_types.h`.

**5.565.2.3** `template<typename _CharT, typename _Traits> typedef  
_Traits::int_type std::istreambuf_iterator<_CharT, _Traits  
>::int_type`

Public typedefs.

Definition at line 62 of file `streambuf_iterator.h`.

**5.565.2.4** `template<typename _CharT, typename _Traits> typedef  
basic_istream<_CharT, _Traits> std::istreambuf_iterator<_CharT,  
_Traits>::istream_type`

Public typedefs.



## **5.565 std::istreambuf\_iterator< \_CharT, \_Traits > Class Template Reference**

Definition at line 64 of file streambuf\_iterator.h.

**5.565.2.5** `typedef input_iterator_tag std::iterator< input_iterator_tag , _CharT ,  
_Traits::off_type , _CharT * , _CharT & >::iterator_category  
[inherited]`

One of the [tag types](#).

Definition at line 122 of file stl\_iterator\_base\_types.h.

**5.565.2.6** `typedef _CharT * std::iterator< input_iterator_tag , _CharT ,  
_Traits::off_type , _CharT * , _CharT & >::pointer [inherited]`

This type represents a pointer-to-value\_type.

Definition at line 128 of file stl\_iterator\_base\_types.h.

**5.565.2.7** `typedef _CharT & std::iterator< input_iterator_tag , _CharT ,  
_Traits::off_type , _CharT * , _CharT & >::reference  
[inherited]`

This type represents a reference-to-value\_type.

Definition at line 130 of file stl\_iterator\_base\_types.h.

**5.565.2.8** `template<typename _CharT , typename _Traits > typedef  
basic_streambuf<_CharT, _Traits> std::istreambuf_iterator<  
_CharT, _Traits >::streambuf_type`

Public typedefs.

Definition at line 63 of file streambuf\_iterator.h.

**5.565.2.9** `template<typename _CharT , typename _Traits > typedef _Traits  
std::istreambuf_iterator< _CharT, _Traits >::traits_type`

Public typedefs.

## **5.565 std::istreambuf\_iterator< \_CharT, \_Traits > Class Template Reference**

Definition at line 61 of file streambuf\_iterator.h.

**5.565.2.10** `typedef _CharT std::iterator< input_iterator_tag , _CharT  
, _Traits::off_type , _CharT * , _CharT & >::value_type  
[inherited]`

The type "pointed to" by the iterator.

Definition at line 124 of file stl\_iterator\_base\_types.h.

### **5.565.3 Constructor & Destructor Documentation**

**5.565.3.1** `template<typename _CharT , typename _Traits > constexpr  
std::istreambuf_iterator< _CharT, _Traits >::istreambuf_iterator (   
 ) throw () [inline]`

Construct end of input stream iterator.

Definition at line 98 of file streambuf\_iterator.h.

**5.565.3.2** `template<typename _CharT , typename _Traits >  
std::istreambuf_iterator< _CharT, _Traits >::istreambuf_iterator (   
 istream_type & __s ) throw () [inline]`

Construct start of input stream iterator.

Definition at line 102 of file streambuf\_iterator.h.

**5.565.3.3** `template<typename _CharT , typename _Traits >  
std::istreambuf_iterator< _CharT, _Traits >::istreambuf_iterator (   
 streambuf_type * __s ) throw () [inline]`

Construct start of streambuf iterator.

Definition at line 106 of file streambuf\_iterator.h.

#### 5.565.4 Member Function Documentation

**5.565.4.1** `template<typename _CharT, typename _Traits> bool  
std::istreambuf_iterator<_CharT, _Traits>::equal ( const  
istreambuf_iterator<_CharT, _Traits> & __b ) const [inline]`

Return true both iterators are end or both are not end.

Definition at line 162 of file `streambuf_iterator.h`.

**5.565.4.2** `template<typename _CharT, typename _Traits> char_type  
std::istreambuf_iterator<_CharT, _Traits>::operator* ( ) const  
[inline]`

Return the current character pointed to by iterator. This returns `/// streambuf.sgetc()`. It cannot be assigned. NB: The result of `/// operator*()` on an end of stream is undefined.

Definition at line 113 of file `streambuf_iterator.h`.

**5.565.4.3** `template<typename _CharT, typename _Traits>  
istreambuf_iterator std::istreambuf_iterator<_CharT, _Traits  
>::operator++ ( int ) [inline]`

Advance the iterator. Calls `streambuf.sbumpc()`.

Definition at line 142 of file `streambuf_iterator.h`.

References `std::basic_streambuf<_CharT, _Traits>::sbumpc()`.

**5.565.4.4** `template<typename _CharT, typename _Traits>  
istreambuf_iterator& std::istreambuf_iterator<_CharT, _Traits  
>::operator++ ( ) [inline]`

Advance the iterator. Calls `streambuf.sbumpc()`.

Definition at line 127 of file `streambuf_iterator.h`.

References `std::basic_streambuf<_CharT, _Traits>::sbumpc()`.

The documentation for this class was generated from the following file:

## 2817

- [streambuf\\_iterator.h](#)

## Reference > Struct Template Reference

### Common iterator class.

Inheritance diagram for `std::iterator<_Category, _Tp, _Distance, _Pointer, _Reference>`:



## Public Types

- typedef \_Distance **difference\_type**
- typedef \_Category **iterator\_category**
- typedef \_Pointer **pointer**
- typedef \_Reference **reference**
- typedef \_Tp **value\_type**

### 5.566.1 Detailed Description

```
template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t,
typename _Pointer = _Tp*, typename _Reference = _Tp&> struct std::iterator<
_Category, _Tp, _Distance, _Pointer, _Reference >
```

Common iterator class. This class does nothing but define nested typedefs. Iterator classes can inherit from this class to save some work. The typedefs are then used in specializations and overloading.

In particular, there are no default implementations of requirements such as `operator++` and the like. (How could there be?)

Definition at line 119 of file stl\_iterator\_base\_types.h.

## 5.566.2 Member Typedef Documentation

**5.566.2.1** `template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename _Reference = _Tp&> typedef _Distance std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::difference_type`

Distance between iterators is represented as this type.

Reimplemented in [std::reverse\\_iterator< \\_Iterator >](#).

Definition at line 126 of file `stl_iterator_base_types.h`.

**5.566.2.2** `template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename _Reference = _Tp&> typedef _Category std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::iterator_category`

One of the [tag types](#).

Definition at line 122 of file `stl_iterator_base_types.h`.

**5.566.2.3** `template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename _Reference = _Tp&> typedef _Pointer std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::pointer`

This type represents a pointer-to-value\_type.

Reimplemented in [std::reverse\\_iterator< \\_Iterator >](#).

Definition at line 128 of file `stl_iterator_base_types.h`.

**5.566.2.4** `template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename _Reference = _Tp&> typedef _Reference std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::reference`

This type represents a reference-to-value\_type.

Reimplemented in [std::reverse\\_iterator< \\_Iterator >](#).

Definition at line 130 of file `stl_iterator_base_types.h`.

```
5.566.2.5 template<typename _Category, typename _Tp, typename _Distance
 = ptrdiff_t, typename _Pointer = _Tp*, typename _Reference
 = _Tp&> typedef _Tp std::iterator< _Category, _Tp, _Distance,
 _Pointer, _Reference >::value_type
```

The type "pointed to" by the iterator.

Definition at line 124 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 5.567 `std::iterator_traits<_Tp * >` Struct Template Reference

Partial specialization for pointer types.

### Public Types

- typedef `ptrdiff_t` **difference\_type**
- typedef [random\\_access\\_iterator\\_tag](#) **iterator\_category**
- typedef `_Tp *` **pointer**
- typedef `_Tp &` **reference**
- typedef `_Tp` **value\_type**

### 5.567.1 Detailed Description

```
template<typename _Tp> struct std::iterator_traits<_Tp * >
```

Partial specialization for pointer types.

Definition at line 176 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 5.568 `std::iterator_traits< const _Tp * >` Struct Template Reference

Partial specialization for const pointer types.

### Public Types

- typedef ptrdiff\_t **difference\_type**
- typedef [random\\_access\\_iterator\\_tag](#) **iterator\_category**
- typedef const \_Tp \* **pointer**
- typedef const \_Tp & **reference**
- typedef \_Tp **value\_type**

#### 5.568.1 Detailed Description

**template<typename \_Tp> struct std::iterator\_traits< const \_Tp \* >**

Partial specialization for const pointer types.

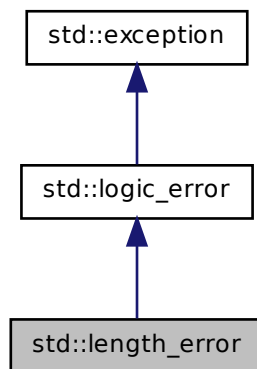
Definition at line 187 of file stl\_iterator\_base\_types.h.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

### 5.569 std::length\_error Class Reference

Inheritance diagram for std::length\_error:



### Public Member Functions

- **length\_error** (const [string](#) &\_\_arg)
- virtual const char \* **what** () const throw ()

#### 5.569.1 Detailed Description

Thrown when an object is constructed that would exceed its maximum permitted size (e.g., a [basic\\_string](#) instance).

Definition at line 91 of file `stdexcept`.

#### 5.569.2 Member Function Documentation

##### 5.569.2.1 virtual const char\* `std::logic_error::what` ( ) const throw () [**virtual**, **inherited**]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future\\_error](#).

The documentation for this class was generated from the following file:

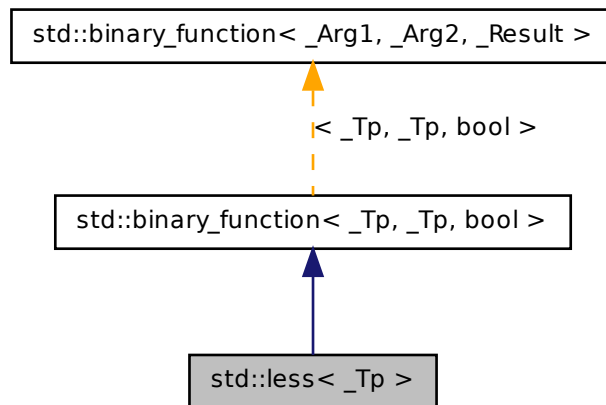
- [stdexcept](#)

### 5.570 `std::less<_Tp>` Struct Template Reference

One of the [comparison functors](#).



Inheritance diagram for `std::less<_Tp>`:



### Public Types

- typedef `_Tp` `first_argument_type`
- typedef `bool` `result_type`
- typedef `_Tp` `second_argument_type`

### Public Member Functions

- `bool operator() (const _Tp &__x, const _Tp &__y) const`

#### 5.570.1 Detailed Description

**template<typename \_Tp> struct `std::less<_Tp>`**

One of the [comparison functors](#).

Definition at line 232 of file `stl_function.h`.

### 5.570.2 Member Typedef Documentation

**5.570.2.1** `typedef _Tp std::binary_function<_Tp, _Tp, bool  
>::first_argument_type [inherited]`

`first_argument_type` is the type of the first argument

Definition at line 118 of file `stl_function.h`.

**5.570.2.2** `typedef bool std::binary_function<_Tp, _Tp, bool >::result_type  
[inherited]`

`result_type` is the return type

Definition at line 124 of file `stl_function.h`.

**5.570.2.3** `typedef _Tp std::binary_function<_Tp, _Tp, bool  
>::second_argument_type [inherited]`

`second_argument_type` is the type of the second argument

Definition at line 121 of file `stl_function.h`.

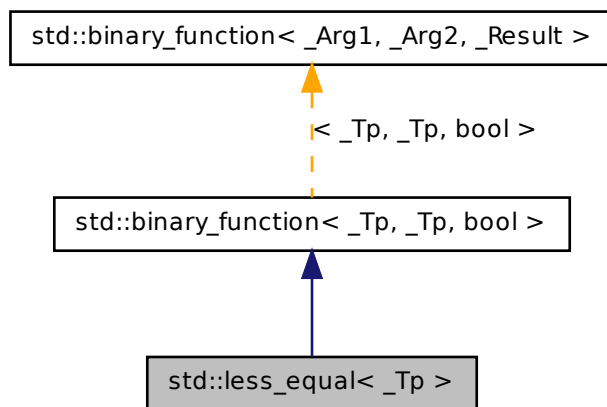
The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.571 `std::less_equal<_Tp>` Struct Template Reference

One of the [comparison functors](#).

Inheritance diagram for std::less\_equal< \_Tp >:



### Public Types

- typedef `_Tp` `first_argument_type`
- typedef `bool` `result_type`
- typedef `_Tp` `second_argument_type`

### Public Member Functions

- `bool operator() (const _Tp &__x, const _Tp &__y) const`

#### 5.571.1 Detailed Description

**template<typename \_Tp> struct std::less\_equal< \_Tp >**

One of the [comparison functions](#).

Definition at line 250 of file `stl_function.h`.

### 5.571.2 Member Typedef Documentation

**5.571.2.1** `typedef _Tp std::binary_function< _Tp , _Tp , bool  
>::first_argument_type [inherited]`

`first_argument_type` is the type of the first argument

Definition at line 118 of file `stl_function.h`.

**5.571.2.2** `typedef bool std::binary_function< _Tp , _Tp , bool >::result_type  
[inherited]`

`result_type` is the return type

Definition at line 124 of file `stl_function.h`.

**5.571.2.3** `typedef _Tp std::binary_function< _Tp , _Tp , bool  
>::second_argument_type [inherited]`

`second_argument_type` is the type of the second argument

Definition at line 121 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## **5.572 std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m > Class Template Reference**

A model of a linear congruential random number generator.

### Public Types

- `typedef _UIntType` [result\\_type](#)

### Public Member Functions

- [linear\\_congruential\\_engine](#) ([result\\_type](#) \_\_s=default\_seed)

- template<typename \_Sseq, typename = typename std::enable\_if<!std::is\_same<\_Sseq, linear\_congruential\_engine>::value>::type>  
[linear\\_congruential\\_engine](#) (\_Sseq &\_\_q)
- void [discard](#) (unsigned long long \_\_z)
- [result\\_type](#) operator() ()
- template<typename \_Sseq >  
[std::enable\\_if< std::is\\_class<\\_Sseq>::value >::type](#) [seed](#) (\_Sseq &\_\_q)
- void [seed](#) ([result\\_type](#) \_\_s=default\_seed)

### Static Public Member Functions

- static constexpr [result\\_type](#) [max](#) ()
- static constexpr [result\\_type](#) [min](#) ()

### Static Public Attributes

- static constexpr [result\\_type](#) [default\\_seed](#)
- static constexpr [result\\_type](#) [increment](#)
- static constexpr [result\\_type](#) [modulus](#)
- static constexpr [result\\_type](#) [multiplier](#)

### Friends

- template<typename \_UIntType1, \_UIntType1 \_\_a1, \_UIntType1 \_\_c1, \_UIntType1 \_\_m1, typename \_CharT, typename \_Traits >  
[std::basic\\_ostream](#)< \_CharT, \_Traits > & [operator<<](#) ([std::basic\\_ostream](#)< \_CharT, \_Traits > &, const [std::linear\\_congruential\\_engine](#)< \_UIntType1, \_\_a1, \_\_c1, \_\_m1 > &)
- bool [operator==](#) (const [linear\\_congruential\\_engine](#) &\_\_lhs, const [linear\\_congruential\\_engine](#) &\_\_rhs)
- template<typename \_UIntType1, \_UIntType1 \_\_a1, \_UIntType1 \_\_c1, \_UIntType1 \_\_m1, typename \_CharT, typename \_Traits >  
[std::basic\\_istream](#)< \_CharT, \_Traits > & [operator>>](#) ([std::basic\\_istream](#)< \_CharT, \_Traits > &, [std::linear\\_congruential\\_engine](#)< \_UIntType1, \_\_a1, \_\_c1, \_\_m1 > &)

### 5.572.1 Detailed Description

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
class std::linear_congruential_engine< _UIntType, __a, __c, __m >
```

A model of a linear congruential random number generator. A random number generator that produces pseudorandom numbers via linear function:

$$x_{i+1} \leftarrow (ax_i + c) \bmod m$$

The template parameter `_UIntType` must be an unsigned integral type large enough to store values up to  $(\text{__m}-1)$ . If the template parameter `__m` is 0, the modulus `__m` used is `std::numeric_limits<_UIntType>::max()` plus 1. Otherwise, the template parameters `__a` and `__c` must be less than `__m`.

The size of the state is 1.

Definition at line 170 of file `random.h`.

### 5.572.2 Member Typedef Documentation

**5.572.2.1** `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> typedef _UIntType std::linear_congruential_engine< _UIntType, __a, __c, __m >::result_type`

The type of the generated random value.

Definition at line 179 of file `random.h`.

### 5.572.3 Constructor & Destructor Documentation

**5.572.3.1** `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> std::linear_congruential_engine< _UIntType, __a, __c, __m >::linear_congruential_engine ( result_type __s = default_seed ) [inline, explicit]`

Constructs a `linear_congruential_engine` random number generator engine with seed `__s`. The default seed value is 1.

#### Parameters

`__s` The initial seed value.

Definition at line 197 of file `random.h`.

References `std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed()`.

**5.572.3.2** `template<typename _UIntType, _UIntType __a, _UIntType  
__c, _UIntType __m> template<typename _Sseq, typename =  
typename std::enable_if<!std::is_same<_Sseq, linear_congruential_  
engine>::value> ::type> std::linear_congruential_engine<  
_UIntType, __a, __c, __m >::linear_congruential_engine ( _Sseq &  
__q ) [inline, explicit]`

Constructs a `linear_congruential_engine` random number generator engine seeded from the seed sequence `__q`.

#### Parameters

`__q` the seed sequence.

Definition at line 210 of file `random.h`.

References `std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed()`.

### 5.572.4 Member Function Documentation

**5.572.4.1** `template<typename _UIntType, _UIntType __a, _UIntType __c,  
_UIntType __m> void std::linear_congruential_engine< _UIntType,  
__a, __c, __m >::discard ( unsigned long long __z ) [inline]`

Discard a sequence of random numbers.

Definition at line 254 of file `random.h`.

**5.572.4.2** `template<typename _UIntType, _UIntType __a, _UIntType  
__c, _UIntType __m> static constexpr result_type  
std::linear_congruential_engine< _UIntType, __a, __c, __m >::max ( )  
[inline, static]`

Gets the largest possible value in the output range.

Definition at line 247 of file `random.h`.

**5.572.4.3** `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> static constexpr result_type  
std::linear_congruential_engine<_UIntType, __a, __c, __m>::min ( ) [inline, static]`

Gets the smallest possible value in the output range.

The minimum depends on the `__c` parameter: if it is zero, the minimum generated must be  $> 0$ , otherwise 0 is allowed.

Definition at line 240 of file `random.h`.

**5.572.4.4** `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> result_type std::linear_congruential_engine<_UIntType, __a, __c, __m>::operator() ( ) [inline]`

Gets the next random number in the sequence.

Definition at line 264 of file `random.h`.

**5.572.4.5** `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> template<typename _Sseq >  
std::enable_if< std::is_class<_Sseq>::value >::type  
std::linear_congruential_engine<_UIntType, __a, __c, __m>::seed ( _Sseq & __q )`

Reseeds the `linear_congruential_engine` random number generator engine sequence using values from the seed sequence `__q`.

#### Parameters

`__q` the seed sequence.

Seeds the LCR engine with a value generated by `__q`.

Definition at line 150 of file `random.tcc`.

References `std::lg()`, and `std::linear_congruential_engine<_UIntType, __a, __c, __m>::seed()`.



**5.572.4.6** `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> void std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed ( result_type __s = default_seed )`

Reseeds the `linear_congruential_engine` random number generator engine sequence to the seed `__s`.

#### Parameters

`__s` The new seed.

Seeds the LCR with integral value `__s`, adjusted so that the ring identity is never a member of the convergence set.

Definition at line 134 of file `random.tcc`.

Referenced by `std::linear_congruential_engine< _UIntType, __a, __c, __m >::linear_congruential_engine()`, and `std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed()`.

### 5.572.5 Friends And Related Function Documentation

**5.572.5.1** `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> template<typename _UIntType1, _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT, typename _Traits> std::basic_ostream< _CharT, _Traits> & operator<< ( std::basic_ostream< _CharT, _Traits> &, const std::linear_congruential_engine< _UIntType1, __a1, __c1, __m1 > & ) [friend]`

Writes the textual representation of the state `x(i)` of `x` to `__os`.

#### Parameters

`__os` The output stream.

`__lcr` A % [linear\\_congruential\\_engine](#) random number generator.

#### Returns

`__os`.

**5.572.5.2** `template<typename _UIntType, _UIntType __a,  
 _UIntType __c, _UIntType __m> bool operator==( const  
 linear_congruential_engine< _UIntType, __a, __c, __m > & __lhs,  
 const linear_congruential_engine< _UIntType, __a, __c, __m > &  
 __rhs ) [friend]`

Compares two linear congruential random number generator objects of the same type for equality.

**Parameters**

`__lhs` A linear congruential random number generator object.  
`__rhs` Another linear congruential random number generator object.

**Returns**

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 282 of file random.h.

**5.572.5.3** `template<typename _UIntType, _UIntType __a, _UIntType __c,  
 _UIntType __m> template<typename _UIntType1, _UIntType1  
 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT  
 , typename _Traits > std::basic_istream<_CharT, _Traits>&  
 operator>> ( std::basic_istream<_CharT, _Traits> & ,  
 std::linear_congruential_engine< _UIntType1, __a1, __c1, __m1 > &  
 ) [friend]`

Sets the state of the engine by reading its textual representation from `__is`.

The textual representation must have been previously written using an output stream whose imbued locale and whose type's template specialization arguments `_CharT` and `_Traits` were the same as those of `__is`.

**Parameters**

`__is` The input stream.  
`__lcr` A % [linear\\_congruential\\_engine](#) random number generator.

**Returns**

`__is`.

### 5.572.6 Member Data Documentation

**5.572.6.1** `template<typename _UIntType, _UIntType __a,  
_UIntType __c, _UIntType __m> constexpr _UIntType  
std::linear_congruential_engine<_UIntType, __a, __c, __m  
>::increment [static]`

An increment.

Definition at line 184 of file `random.h`.

**5.572.6.2** `template<typename _UIntType, _UIntType __a,  
_UIntType __c, _UIntType __m> constexpr _UIntType  
std::linear_congruential_engine<_UIntType, __a, __c, __m  
>::modulus [static]`

The modulus.

Definition at line 186 of file `random.h`.

**5.572.6.3** `template<typename _UIntType, _UIntType __a,  
_UIntType __c, _UIntType __m> constexpr _UIntType  
std::linear_congruential_engine<_UIntType, __a, __c, __m  
>::multiplier [static]`

The multiplier.

Definition at line 182 of file `random.h`.

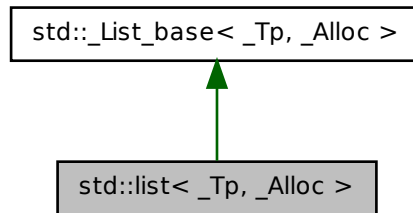
The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.573 `std::list<_Tp, _Alloc>` Class Template Reference

A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.

Inheritance diagram for `std::list< _Tp, _Alloc >`:



### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_List_const_iterator< _Tp >` **const\_iterator**
- typedef `_Tp_alloc_type::const_pointer` **const\_pointer**
- typedef `_Tp_alloc_type::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `ptrdiff_t` **difference\_type**
- typedef `_List_iterator< _Tp >` **iterator**
- typedef `_Tp_alloc_type::pointer` **pointer**
- typedef `_Tp_alloc_type::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `size_t` **size\_type**
- typedef `_Tp` **value\_type**

### Public Member Functions

- `list()`
- `list(const allocator_type &__a)`
- `list(size_type __n, const value_type &__value, const allocator_type &__a=allocator_type())`
- `template<typename _InputIterator >`  
`list(_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())`
- `list(const list &__x)`
- `list(size_type __n)`

- [list](#) ([list](#) &&\_\_x)
- [list](#) ([initializer\\_list](#)< value\_type > \_\_l, const allocator\_type &\_\_a=allocator\_type())
- void [assign](#) (size\_type \_\_n, const value\_type &\_\_val)
- template<typename \_InputIterator >  
void [assign](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void [assign](#) ([initializer\\_list](#)< value\_type > \_\_l)
- reference [back](#) ()
- const\_reference [back](#) () const
- [iterator](#) [begin](#) ()
- [const\\_iterator](#) [begin](#) () const
- [const\\_iterator](#) [cbegin](#) () const
- [const\\_iterator](#) [cend](#) () const
- void [clear](#) ()
- [const\\_reverse\\_iterator](#) [crbegin](#) () const
- [const\\_reverse\\_iterator](#) [crend](#) () const
- template<typename... \_Args>  
[iterator](#) [emplace](#) ([iterator](#) \_\_position, \_Args &&...\_\_args)
- template<typename... \_Args>  
void [emplace\\_back](#) (\_Args &&...\_\_args)
- template<typename... \_Args>  
void [emplace\\_front](#) (\_Args &&...\_\_args)
- bool [empty](#) () const
- [iterator](#) [end](#) ()
- [const\\_iterator](#) [end](#) () const
- [iterator](#) [erase](#) ([iterator](#) \_\_position)
- [iterator](#) [erase](#) ([iterator](#) \_\_first, [iterator](#) \_\_last)
- reference [front](#) ()
- const\_reference [front](#) () const
- allocator\_type [get\\_allocator](#) () const
- [iterator](#) [insert](#) ([iterator](#) \_\_position, const value\_type &\_\_x)
- [iterator](#) [insert](#) ([iterator](#) \_\_position, value\_type &&\_\_x)
- void [insert](#) ([iterator](#) \_\_p, [initializer\\_list](#)< value\_type > \_\_l)
- void [insert](#) ([iterator](#) \_\_position, size\_type \_\_n, const value\_type &\_\_x)
- template<typename \_InputIterator >  
void [insert](#) ([iterator](#) \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- size\_type [max\\_size](#) () const
- void [merge](#) ([list](#) &&\_\_x)
- void [merge](#) ([list](#) &\_\_x)
- template<typename \_StrictWeakOrdering >  
void [merge](#) ([list](#) &&, \_StrictWeakOrdering)
- template<typename \_StrictWeakOrdering >  
void [merge](#) ([list](#) &\_\_x, \_StrictWeakOrdering \_\_comp)

- `list & operator=` (const `list` &\_\_x)
- `list & operator=` (list &&\_\_x)
- `list & operator=` (initializer\_list< value\_type > \_\_l)
- void `pop_back` ()
- void `pop_front` ()
- void `push_back` (const value\_type &\_\_x)
- void `push_back` (value\_type &&\_\_x)
- void `push_front` (value\_type &&\_\_x)
- void `push_front` (const value\_type &\_\_x)
- `reverse_iterator` `rbegin` ()
- `const_reverse_iterator` `rbegin` () const
- void `remove` (const \_Tp &\_\_value)
- template<typename \_Predicate >  
void `remove_if` (\_Predicate)
- `reverse_iterator` `rend` ()
- `const_reverse_iterator` `rend` () const
- void `resize` (size\_type \_\_new\_size)
- void `resize` (size\_type \_\_new\_size, const value\_type &\_\_x)
- void `reverse` ()
- size\_type `size` () const
- template<typename \_StrictWeakOrdering >  
void `sort` (\_StrictWeakOrdering)
- void `sort` ()
- void `splice` (iterator \_\_position, list &\_\_x, iterator \_\_i)
- void `splice` (iterator \_\_position, list &&\_\_x, iterator \_\_i)
- void `splice` (iterator \_\_position, list &\_\_x)
- void `splice` (iterator \_\_position, list &\_\_x, iterator \_\_first, iterator \_\_last)
- void `splice` (iterator \_\_position, list &&\_\_x, iterator \_\_first, iterator \_\_last)
- void `splice` (iterator \_\_position, list &&\_\_x)
- void `swap` (list &\_\_x)
- template<typename \_BinaryPredicate >  
void `unique` (\_BinaryPredicate)
- void `unique` ()

### Protected Types

- typedef `_List_node`< \_Tp > `_Node`
- typedef `_Alloc::template rebind`< `_List_node`< \_Tp > ::other `_Node_alloc_type`

**Protected Member Functions**

- `template<typename _Integer>`  
`void _M_assign_dispatch (_Integer __n, _Integer __val, __true_type)`
- `template<typename _InputIterator>`  
`void _M_assign_dispatch (_InputIterator __first, _InputIterator __last, __false_type)`
- `void _M_check_equal_allocators (list &__x)`
- `void _M_clear ()`
- `template<typename... _Args>`  
`_Node * _M_create_node (_Args &&...__args)`
- `void _M_default_append (size_type __n)`
- `void _M_default_initialize (size_type __n)`
- `void _M_erase (iterator __position)`
- `void _M_fill_assign (size_type __n, const value_type &__val)`
- `void _M_fill_initialize (size_type __n, const value_type &__x)`
- `_List_node<_Tp> * _M_get_node ()`
- `_Node_alloc_type & _M_get_Node_allocator ()`
- `const _Node_alloc_type & _M_get_Node_allocator () const`
- `_Tp_alloc_type _M_get_Tp_allocator () const`
- `void _M_init ()`
- `template<typename _InputIterator>`  
`void _M_initialize_dispatch (_InputIterator __first, _InputIterator __last, __false_type)`
- `template<typename _Integer>`  
`void _M_initialize_dispatch (_Integer __n, _Integer __x, __true_type)`
- `template<typename... _Args>`  
`void _M_insert (iterator __position, _Args &&...__args)`
- `void _M_put_node (_List_node<_Tp> *__p)`
- `void _M_transfer (iterator __position, iterator __first, iterator __last)`

**Protected Attributes**

- `_List_impl _M_impl`

**5.573.1 Detailed Description**

`template<typename _Tp, typename _Alloc = std::allocator<_Tp>> class std::list<_Tp, _Alloc>`

A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence. Meets the requirements of a [container](#),

a [reversible container](#), and a [sequence](#), including the [optional sequence requirements](#) with the exception of `at` and `operator[]`.

This is a *doubly linked* list. Traversal up and down the list requires linear time, but adding and removing elements (or *nodes*) is done in constant time, regardless of where the change takes place. Unlike [std::vector](#) and [std::deque](#), random-access iterators are not provided, so subscripting ( `[]` ) access is not allowed. For algorithms which only need sequential access, this lack makes no difference.

Also unlike the other standard containers, [std::list](#) provides specialized algorithms unique to linked lists, such as splicing, sorting, and in-place reversal.

A couple points on memory allocation for `list<Tp>`:

First, we never actually allocate a `Tp`, we allocate `List_node<Tp>`'s and trust [20.1.5]/4 to DTRT. This is to ensure that after elements from `list<X, Alloc1>` are spliced into `list<X, Alloc2>`, destroying the memory of the second list is a valid operation, i.e., `Alloc1` giveth and `Alloc2` taketh away.

Second, a list conceptually represented as

```
A <----> B <----> C <----> D
```

is actually circular; a link exists between A and D. The list class holds (as its only data member) a private `list::iterator` pointing to *D*, not to *A*! To get to the head of the list, we start at the tail and move forward by one. When this member iterator's `next/previous` pointers refer to itself, the list is empty.

Definition at line 429 of file `stl_list.h`.

## 5.573.2 Constructor & Destructor Documentation

**5.573.2.1** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
std::list<_Tp, _Alloc>::list ( ) [inline]`

Default constructor creates no elements.

Definition at line 512 of file `stl_list.h`.

**5.573.2.2** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
std::list<_Tp, _Alloc>::list ( const allocator_type & __a )  
[inline, explicit]`

Creates a list with no elements.



**Parameters**

*a* An allocator object.

Definition at line 520 of file `stl_list.h`.

**5.573.2.3** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
std::list<_Tp, _Alloc>::list ( size_type __n ) [inline,  
explicit]`

Creates a list with default constructed elements.

**Parameters**

*n* The number of elements to initially create.

This constructor fills the list with *n* default constructed elements.

Definition at line 532 of file `stl_list.h`.

**5.573.2.4** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
std::list<_Tp, _Alloc>::list ( size_type __n, const value_type &  
__value, const allocator_type & __a = allocator_type() )  
[inline]`

Creates a list with copies of an exemplar element.

**Parameters**

*n* The number of elements to initially create.

*value* An element to copy.

*a* An allocator object.

This constructor fills the list with *n* copies of *value*.

Definition at line 544 of file `stl_list.h`.

**5.573.2.5** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
std::list<_Tp, _Alloc>::list ( const list<_Tp, _Alloc> & __x )  
[inline]`

List copy constructor.

**Parameters**

*x* A list of identical element and allocator types.

The newly-created list uses a copy of the allocation object used by *x*.

Definition at line 571 of file `stl_list.h`.

References `std::list<_Tp, _Alloc>::begin()`, and `std::list<_Tp, _Alloc>::end()`.

```
5.573.2.6 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
 std::list<_Tp, _Alloc>::list (list<_Tp, _Alloc> && __x)
 [inline]
```

List move constructor.

**Parameters**

*x* A list of identical element and allocator types.

The newly-created list contains the exact contents of *x*. The contents of *x* are a valid, but unspecified list.

Definition at line 583 of file `stl_list.h`.

```
5.573.2.7 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
 std::list<_Tp, _Alloc>::list (initializer_list<value_type> __l,
 const allocator_type & __a = allocator_type()) [inline]
```

Builds a list from an [initializer\\_list](#).

**Parameters**

*l* An [initializer\\_list](#) of `value_type`.

*a* An allocator object.

Create a list consisting of copies of the elements in the [initializer\\_list](#) *l*. This is linear in `l.size()`.

Definition at line 594 of file `stl_list.h`.

**5.573.2.8** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
 template<typename _InputIterator > std::list< _Tp, _Alloc >::list (  
 _InputIterator __first, _InputIterator __last, const allocator_type &  
 __a = allocator_type() ) [inline]`

Builds a list from a range.

#### Parameters

*first* An input iterator.

*last* An input iterator.

*a* An allocator object.

Create a list consisting of copies of the elements from [*first*,*last*). This is linear in N (where N is distance(*first*,*last*)).

Definition at line 611 of file stl\_list.h.

### 5.573.3 Member Function Documentation

**5.573.3.1** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
 template<typename... _Args> _Node* std::list< _Tp, _Alloc  
 >::M_create_node ( _Args &&... __args ) [inline,  
 protected]`

#### Parameters

*x* An instance of user data.

Allocates space for a new node and constructs a copy of *x* in it.

Definition at line 489 of file stl\_list.h.

Referenced by std::list< \_Tp, \_Alloc >::emplace(), and std::list< \_Tp, \_Alloc >::insert().

**5.573.3.2** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
 void std::list< _Tp, _Alloc >::assign ( size_type __n, const  
 value_type & __val ) [inline]`

Assigns a given value to a list.

**Parameters**

- n* Number of elements to be assigned.
- val* Value to be assigned.

This function fills a list with *n* copies of the given value. Note that the assignment completely changes the list and that the resulting list's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 682 of file stl\_list.h.

Referenced by std::list< \_Tp, \_Alloc >::operator=().

```
5.573.3.3 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator > void std::list< _Tp, _Alloc
>::assign (_InputIterator __first, _InputIterator __last)
[inline]
```

Assigns a range to a list.

**Parameters**

- first* An input iterator.
- last* An input iterator.

This function fills a list with copies of the elements in the range [*first*,*last*).

Note that the assignment completely changes the list and that the resulting list's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 699 of file stl\_list.h.

```
5.573.3.4 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::assign (initializer_list< value_type >
__l) [inline]
```

Assigns an [initializer\\_list](#) to a list.

**Parameters**

- l* An [initializer\\_list](#) of value\_type.

Replace the contents of the list with copies of the elements in the [initializer\\_list](#) *l*. This is linear in *l.size()*.

Definition at line 715 of file `stl_list.h`.

References `std::list<_Tp, _Alloc>::assign()`.

Referenced by `std::list<_Tp, _Alloc>::assign()`.

#### 5.573.3.5 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::list<_Tp, _Alloc>::back ( ) [inline]`

Returns a read/write reference to the data at the last element of the list.

Definition at line 915 of file `stl_list.h`.

References `std::list<_Tp, _Alloc>::end()`.

#### 5.573.3.6 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::list<_Tp, _Alloc>::back ( ) const [inline]`

Returns a read-only (constant) reference to the data at the last element of the list.

Definition at line 927 of file `stl_list.h`.

References `std::list<_Tp, _Alloc>::end()`.

#### 5.573.3.7 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::list<_Tp, _Alloc>::begin ( ) [inline]`

Returns a read/write iterator that points to the first element in the list. Iteration is done in ordinary element order.

Definition at line 730 of file `stl_list.h`.

Referenced by `std::list<_Tp, _Alloc>::crend()`, `std::list<_Tp, _Alloc>::front()`, `std::list<_Tp, _Alloc>::list()`, `std::list<_Tp, _Alloc>::merge()`, `std::list<_Tp, _Alloc>::operator=()`, `std::operator==(std::list<_Tp, _Alloc>, std::list<_Tp, _Alloc>)`, `std::list<_Tp, _Alloc>::pop_front()`, `std::list<_Tp, _Alloc>::push_front()`, `std::list<_Tp, _Alloc>::remove()`, `std::list<_Tp, _Alloc>::remove_if()`, `std::list<_Tp, _Alloc>::rend()`, `std::list<_Tp, _Alloc>::resize()`, `std::list<_Tp, _Alloc>::size()`, `std::list<_Tp, _Alloc>::sort()`, `std::list<_Tp, _Alloc>::splice()`, and `std::list<_Tp, _Alloc>::unique()`.

#### 5.573.3.8 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::list<_Tp, _Alloc>::begin ( ) const [inline]`

Returns a read-only (constant) iterator that points to the first element in the list. Iteration is done in ordinary element order.

Definition at line 739 of file `stl_list.h`.

**5.573.3.9** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_iterator std::list<_Tp, _Alloc>::cbegin ( ) const [inline]`

Returns a read-only (constant) iterator that points to the first element in the list. Iteration is done in ordinary element order.

Definition at line 803 of file `stl_list.h`.

**5.573.3.10** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_iterator std::list<_Tp, _Alloc>::cend ( ) const [inline]`

Returns a read-only (constant) iterator that points one past the last element in the list. Iteration is done in ordinary element order.

Definition at line 812 of file `stl_list.h`.

**5.573.3.11** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::list<_Tp, _Alloc>::clear ( ) [inline]`

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1195 of file `stl_list.h`.

Referenced by `std::list<_Tp, _Alloc>::operator=()`.

**5.573.3.12** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_reverse_iterator std::list<_Tp, _Alloc>::crbegin ( ) const  
[inline]`

Returns a read-only (constant) reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

Definition at line 821 of file `stl_list.h`.

References `std::list<_Tp, _Alloc>::end()`.

**5.573.3.13** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_reverse_iterator std::list<_Tp, _Alloc>::crend ( ) const  
[inline]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

Definition at line 830 of file `stl_list.h`.

References std::list< \_Tp, \_Alloc >::begin().

**5.573.3.14** `template<typename _Tp, typename _Alloc > template<typename...  
_Args> list< _Tp, _Alloc >::iterator list::emplace ( iterator  
_position, _Args &&... _args )`

Constructs object in list before specified iterator.

#### Parameters

*position* A const\_iterator into the list.

*args* Arguments.

#### Returns

An iterator that points to the inserted data.

This function will insert an object of type T constructed with T(std::forward<Args>(args)...) before the specified location. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 89 of file list.tcc.

References std::list< \_Tp, \_Alloc >::\_M\_create\_node().

Referenced by std::list< \_Tp, \_Alloc >::insert().

**5.573.3.15** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
bool std::list< _Tp, _Alloc >::empty ( ) const [inline]`

Returns true if the list is empty. (Thus [begin\(\)](#) would equal [end\(\)](#).)

Definition at line 840 of file stl\_list.h.

Referenced by std::list< \_Tp, \_Alloc >::sort(), and std::list< \_Tp, \_Alloc >::splice().

**5.573.3.16** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
iterator std::list< _Tp, _Alloc >::end ( ) [inline]`

Returns a read/write iterator that points one past the last element in the list. Iteration is done in ordinary element order.

Definition at line 748 of file stl\_list.h.

Referenced by std::list< \_Tp, \_Alloc >::back(), std::list< \_Tp, \_Alloc >::cbegin(), std::list< \_Tp, \_Alloc >::list(), std::list< \_Tp, \_Alloc >::merge(), std::list< \_Tp, \_Alloc >::operator=(), std::operator==( ), std::list< \_Tp, \_Alloc >::push\_back(), std::list< \_Tp, \_Alloc >::rbegin(), std::list< \_Tp, \_Alloc >::remove(), std::list< \_Tp, \_Alloc >::remove\_if(), std::list< \_Tp, \_Alloc >::resize(), std::list< \_Tp, \_Alloc >::size(), std::list< \_Tp, \_Alloc >::splice(), and std::list< \_Tp, \_Alloc >::unique().

#### 5.573.3.17 template<typename \_Tp, typename \_Alloc = std::allocator<\_Tp>> const\_iterator std::list< \_Tp, \_Alloc >::end ( ) const [inline]

Returns a read-only (constant) iterator that points one past the last element in the list. Iteration is done in ordinary element order.

Definition at line 757 of file stl\_list.h.

#### 5.573.3.18 template<typename \_Tp, typename \_Alloc > list< \_Tp, \_Alloc >::iterator list::erase ( iterator \_\_position )

Remove element at given position.

##### Parameters

*position* Iterator pointing to element to be erased.

##### Returns

An iterator pointing to the next element (or [end\(\)](#)).

This function will erase the element at the given position and thus shorten the list by one.

Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 110 of file list.tcc.

Referenced by std::list< \_Tp, \_Alloc >::erase(), std::list< \_Tp, \_Alloc >::operator=(), and std::list< \_Tp, \_Alloc >::resize().



**5.573.3.19** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
iterator std::list< _Tp, _Alloc >::erase ( iterator __first, iterator  
__last ) [inline]`

Remove a range of elements.

#### Parameters

*first* Iterator pointing to the first element to be erased.

*last* Iterator pointing to one past the last element to be erased.

#### Returns

An iterator pointing to the element pointed to by *last* prior to erasing (or `end()`).

This function will erase the elements in the range [first,last) and shorten the list accordingly.

This operation is linear time in the size of the range and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1160 of file `stl_list.h`.

References `std::list< _Tp, _Alloc >::erase()`.

**5.573.3.20** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
reference std::list< _Tp, _Alloc >::front ( ) [inline]`

Returns a read/write reference to the data at the first element of the list.

Definition at line 899 of file `stl_list.h`.

References `std::list< _Tp, _Alloc >::begin()`.

**5.573.3.21** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_reference std::list< _Tp, _Alloc >::front ( ) const  
[inline]`

Returns a read-only (constant) reference to the data at the first element of the list.

Definition at line 907 of file `stl_list.h`.

References `std::list< _Tp, _Alloc >::begin()`.

**5.573.3.22** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
allocator_type std::list<_Tp, _Alloc>::get_allocator ( ) const  
[inline]`

Get a copy of the memory allocation object.

Reimplemented from `std::_List_base<_Tp, _Alloc>`.

Definition at line 721 of file `stl_list.h`.

**5.573.3.23** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
iterator std::list<_Tp, _Alloc>::insert ( iterator __position,  
value_type && __x ) [inline]`

Inserts given rvalue into list before specified iterator.

#### Parameters

*position* An iterator into the list.

*x* Data to be inserted.

#### Returns

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 1061 of file `stl_list.h`.

References `std::list<_Tp, _Alloc>::emplace()`.

**5.573.3.24** `template<typename _Tp, typename _Alloc> list<_Tp, _Alloc  
>::iterator list::insert ( iterator __position, const value_type &  
__x )`

Inserts given value into list before specified iterator.

#### Parameters

*position* An iterator into the list.

*x* Data to be inserted.

**Returns**

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 100 of file list.tcc.

References std::list< \_Tp, \_Alloc >::\_M\_create\_node().

Referenced by std::list< \_Tp, \_Alloc >::operator=(), and std::list< \_Tp, \_Alloc >::resize().

**5.573.3.25** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::list< _Tp, _Alloc >::insert ( iterator __p, initializer_list<  
value_type > __l ) [inline]`

Inserts the contents of an [initializer\\_list](#) into list before specified iterator.

**Parameters**

*p* An iterator into the list.

*l* An [initializer\\_list](#) of value\_type.

This function will insert copies of the data in the [initializer\\_list](#) *l* into the list before the location specified by *p*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 1078 of file stl\_list.h.

References std::list< \_Tp, \_Alloc >::insert().

Referenced by std::list< \_Tp, \_Alloc >::insert().

**5.573.3.26** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::list< _Tp, _Alloc >::insert ( iterator __position, size_type  
__n, const value_type & __x ) [inline]`

Inserts a number of copies of given data into the list.

**Parameters**

*position* An iterator into the list.

*n* Number of elements to be inserted.

*x* Data to be inserted.

This function will insert a specified number of copies of the given data before the location specified by *position*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 1095 of file stl\_list.h.

References std::list< \_Tp, \_Alloc >::splice().

```
5.573.3.27 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator > void std::list< _Tp,
_Alloc >::insert (iterator __position, _InputIterator __first,
_InputIterator __last) [inline]
```

Inserts a range into the list.

#### Parameters

*position* An iterator into the list.

*first* An input iterator.

*last* An input iterator.

This function will insert copies of the data in the range [*first*,*last*) into the list before the location specified by *position*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 1116 of file stl\_list.h.

References std::list< \_Tp, \_Alloc >::splice().

```
5.573.3.28 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::list< _Tp, _Alloc >::max_size () const [inline]
```

Returns the [size\(\)](#) of the largest possible list.

Definition at line 850 of file stl\_list.h.

**5.573.3.29** `template<typename _Tp, typename _Alloc > void list::merge ( list< _Tp, _Alloc > && __x )`

Merge sorted lists.

#### Parameters

*x* Sorted list to merge.

Assumes that both *x* and this list are sorted according to operator<(). Merges elements of *x* into this list in sorted order, leaving *x* empty when complete. Elements in this list precede elements in *x* that are equal.

Definition at line 289 of file list.tcc.

References std::list< \_Tp, \_Alloc >::begin(), std::begin(), std::list< \_Tp, \_Alloc >::end(), and std::end().

Referenced by std::list< \_Tp, \_Alloc >::sort().

**5.573.3.30** `template<typename _Tp, typename _Alloc > template<typename _StrictWeakOrdering > void list::merge ( list< _Tp, _Alloc > && __x, _StrictWeakOrdering __comp )`

Merge sorted lists according to comparison function.

#### Parameters

*x* Sorted list to merge.

*StrictWeakOrdering* Comparison function defining sort order.

Assumes that both *x* and this list are sorted according to StrictWeakOrdering. Merges elements of *x* into this list in sorted order, leaving *x* empty when complete. Elements in this list precede elements in *x* that are equivalent according to StrictWeakOrdering().

Definition at line 323 of file list.tcc.

References std::list< \_Tp, \_Alloc >::begin(), std::begin(), std::list< \_Tp, \_Alloc >::end(), and std::end().

**5.573.3.31** `template<typename _Tp, typename _Alloc > list< _Tp, _Alloc > & list::operator= ( const list< _Tp, _Alloc > & __x )`

List assignment operator.

No explicit dtor needed as the `_Base` dtor takes care of things. The `_Base` dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

#### Parameters

*x* A list of identical element and allocator types.

All the elements of *x* are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 186 of file `list.tcc`.

References `std::list< _Tp, _Alloc >::begin()`, `std::list< _Tp, _Alloc >::end()`, `std::list< _Tp, _Alloc >::erase()`, and `std::list< _Tp, _Alloc >::insert()`.

**5.573.3.32** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
list& std::list< _Tp, _Alloc >::operator= ( initializer_list<  
value_type > __l ) [inline]`

List initializer list assignment operator.

#### Parameters

*l* An [initializer\\_list](#) of `value_type`.

Replace the contents of the list with copies of the elements in the [initializer\\_list](#) *l*. This is linear in `l.size()`.

Definition at line 664 of file `stl_list.h`.

References `std::list< _Tp, _Alloc >::assign()`.

**5.573.3.33** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
list& std::list< _Tp, _Alloc >::operator= ( list< _Tp, _Alloc > &&  
__x ) [inline]`

List move assignment operator.

#### Parameters

*x* A list of identical element and allocator types.

The contents of *x* are moved into this list (without copying). *x* is a valid, but unspecified list

Definition at line 647 of file `stl_list.h`.

References `std::list< _Tp, _Alloc >::clear()`, and `std::list< _Tp, _Alloc >::swap()`.

**5.573.3.34** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::list< _Tp, _Alloc >::pop_back ( ) [inline]`

Removes last element.

This is a typical stack operation. It shrinks the list by one. Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before `pop_back()` is called.

Definition at line 1013 of file `stl_list.h`.

**5.573.3.35** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::list< _Tp, _Alloc >::pop_front ( ) [inline]`

Removes first element.

This is a typical stack operation. It shrinks the list by one. Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop_front()` is called.

Definition at line 973 of file `stl_list.h`.

References `std::list< _Tp, _Alloc >::begin()`.

**5.573.3.36** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::list< _Tp, _Alloc >::push_back ( const value_type & __x  
) [inline]`

Add data to the end of the list.

**Parameters**

*x* Data to be added.

This is a typical stack operation. The function creates an element at the end of the list and assigns the given data to it. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 987 of file stl\_list.h.

References std::list< \_Tp, \_Alloc >::end().

```
5.573.3.37 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
 void std::list< _Tp, _Alloc >::push_front (const value_type & __x
) [inline]
```

Add data to the front of the list.

**Parameters**

*x* Data to be added.

This is a typical stack operation. The function creates an element at the front of the list and assigns the given data to it. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 946 of file stl\_list.h.

References std::list< \_Tp, \_Alloc >::begin().

```
5.573.3.38 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
 reverse_iterator std::list< _Tp, _Alloc >::rbegin () [inline]
```

Returns a read/write reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

Definition at line 766 of file stl\_list.h.

References std::list< \_Tp, \_Alloc >::end().

```
5.573.3.39 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
 const_reverse_iterator std::list< _Tp, _Alloc >::rbegin () const
 [inline]
```

Returns a read-only (constant) reverse iterator that points to the last element in the list. Iteration is done in reverse element order.



Definition at line 775 of file stl\_list.h.

References std::list< \_Tp, \_Alloc >::end().

**5.573.3.40** `template<typename _Tp, typename _Alloc > void list::remove (const _Tp & __value )`

Remove all elements equal to value.

#### Parameters

*value* The value to remove.

Removes every element in the list equal to *value*. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 240 of file list.tcc.

References std::list< \_Tp, \_Alloc >::begin(), and std::list< \_Tp, \_Alloc >::end().

**5.573.3.41** `template<typename _Tp , typename _Alloc > template<typename _Predicate > void list::remove_if ( _Predicate __pred )`

Remove all elements satisfying a predicate.

#### Parameters

*Predicate* Unary predicate function or object.

Removes every element in the list for which the predicate returns true. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 393 of file list.tcc.

References std::list< \_Tp, \_Alloc >::begin(), and std::list< \_Tp, \_Alloc >::end().

**5.573.3.42** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_reverse_iterator std::list< _Tp, _Alloc >::rend ( ) const  
[inline]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

Definition at line 793 of file `std_list.h`.

References `std::list< _Tp, _Alloc >::begin()`.

**5.573.3.43** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
reverse_iterator std::list< _Tp, _Alloc >::rend ( ) [inline]`

Returns a read/write reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

Definition at line 784 of file `std_list.h`.

References `std::list< _Tp, _Alloc >::begin()`.

**5.573.3.44** `template<typename _Tp , typename _Alloc > void list::resize (   
size_type __new_size, const value_type & __x )`

Resizes the list to the specified number of elements.

#### Parameters

*new\_size* Number of elements the list should contain.

*x* Data with which new elements should be populated.

This function will resize the list to the specified number of elements. If the number is smaller than the list's current size the list is truncated, otherwise the list is extended and new elements are populated with given data.

Definition at line 155 of file `list.tcc`.

References `std::list< _Tp, _Alloc >::begin()`, `std::list< _Tp, _Alloc >::end()`, `std::list< _Tp, _Alloc >::erase()`, and `std::list< _Tp, _Alloc >::insert()`.

**5.573.3.45** `template<typename _Tp , typename _Alloc > void list::resize (   
size_type __new_size )`

Resizes the list to the specified number of elements.

**Parameters**

*new\_size* Number of elements the list should contain.

This function will resize the list to the specified number of elements. If the number is smaller than the list's current size the list is truncated, otherwise default constructed elements are appended.

Definition at line 140 of file `list.tcc`.

References `std::list< _Tp, _Alloc >::begin()`, `std::list< _Tp, _Alloc >::end()`, and `std::list< _Tp, _Alloc >::erase()`.

**5.573.3.46** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::list< _Tp, _Alloc >::reverse ( ) [inline]`

Reverse the elements in list.

Reverse the order of elements in the list in linear time.

Definition at line 1415 of file `stl_list.h`.

**5.573.3.47** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
size_type std::list< _Tp, _Alloc >::size ( ) const [inline]`

Returns the number of elements in the list.

Definition at line 845 of file `stl_list.h`.

References `std::list< _Tp, _Alloc >::begin()`, `std::distance()`, and `std::list< _Tp, _Alloc >::end()`.

**5.573.3.48** `template<typename _Tp, typename _Alloc > void list::sort ( )`

Sort the elements.

Sorts the elements of this list in  $N \log N$  time. Equivalent elements remain in list order.

Definition at line 355 of file `list.tcc`.

References `std::list< _Tp, _Alloc >::begin()`, `std::list< _Tp, _Alloc >::empty()`, `std::list< _Tp, _Alloc >::merge()`, `std::list< _Tp, _Alloc >::splice()`, and `std::list< _Tp, _Alloc >::swap()`.

**5.573.3.49** `template<typename _Tp, typename _Alloc > template<typename  
_StrictWeakOrdering > void list::sort ( _StrictWeakOrdering  
__comp )`

Sort the elements according to comparison function.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

Definition at line 432 of file list.tcc.

References `std::list< _Tp, _Alloc >::begin()`, `std::list< _Tp, _Alloc >::empty()`, `std::list< _Tp, _Alloc >::merge()`, `std::list< _Tp, _Alloc >::splice()`, and `std::list< _Tp, _Alloc >::swap()`.

**5.573.3.50** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::list< _Tp, _Alloc >::splice ( iterator __position, list<  
_Tp, _Alloc > && __x, iterator __first, iterator __last )  
[inline]`

Insert range from another list.

#### Parameters

*position* Iterator referencing the element to insert before.

*x* Source list.

*first* Iterator referencing the start of range in *x*.

*last* Iterator referencing the end of range in *x*.

Removes elements in the range `[first,last)` and inserts them before *position* in constant time.

Undefined if *position* is in `[first,last)`.

Definition at line 1281 of file stl\_list.h.

**5.573.3.51** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::list< _Tp, _Alloc >::splice ( iterator __position, list<  
_Tp, _Alloc > && __x, iterator __i ) [inline]`

Insert element from another list.

**Parameters**

*position* Iterator referencing the element to insert before.

*x* Source list.

*i* Iterator referencing the element to move.

Removes the element in list *x* referenced by *i* and inserts it into the current list before *position*.

Definition at line 1245 of file stl\_list.h.

```
5.573.3.52 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::splice (iterator __position, list<
 _Tp, _Alloc > && __x) [inline]
```

Insert contents of another list.

**Parameters**

*position* Iterator referencing the element to insert before.

*x* Source list.

The elements of *x* are inserted in constant time in front of the element referenced by *position*. *x* becomes an empty list.

Requires this != *x*.

Definition at line 1215 of file stl\_list.h.

References std::list< \_Tp, \_Alloc >::begin(), std::list< \_Tp, \_Alloc >::empty(), and std::list< \_Tp, \_Alloc >::end().

Referenced by std::list< \_Tp, \_Alloc >::insert(), and std::list< \_Tp, \_Alloc >::sort().

```
5.573.3.53 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::swap (list< _Tp, _Alloc > & __x)
[inline]
```

Swaps data with another list.

**Parameters**

*x* A list of the same element and allocator types.

This exchanges the elements between two lists in constant time. Note that the global `std::swap()` function is specialized such that `std::swap(l1,l2)` will feed to this function.

Definition at line 1177 of file `stl_list.h`.

Referenced by `std::list< _Tp, _Alloc >::operator=()`, `std::list< _Tp, _Alloc >::sort()`, and `std::swap()`.

#### 5.573.3.54 `template<typename _Tp, typename _Alloc > void list::unique ( )`

Remove consecutive duplicate elements.

For each consecutive set of elements with the same value, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 268 of file `list.tcc`.

References `std::list< _Tp, _Alloc >::begin()`, and `std::list< _Tp, _Alloc >::end()`.

#### 5.573.3.55 `template<typename _Tp, typename _Alloc > template<typename _BinaryPredicate > void list::unique ( _BinaryPredicate __binary_pred )`

Remove consecutive elements satisfying a predicate.

##### Parameters

***BinaryPredicate*** Binary predicate function or object.

For each consecutive set of elements `[first,last)` that satisfy `predicate(first,i)` where `i` is an iterator in `[first,last)`, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 411 of file `list.tcc`.

References `std::list< _Tp, _Alloc >::begin()`, and `std::list< _Tp, _Alloc >::end()`.

The documentation for this class was generated from the following files:

- [stl\\_list.h](#)
- [list.tcc](#)

## 5.574 std::locale Class Reference

Container class for localization functionality.

The locale class is first a class wrapper for C library locales. It is also an extensible container for user-defined localization. A locale is a collection of facets that implement various localization features such as money, time, and number printing.

### Classes

- class [facet](#)

*Localization functionality base class.*

*The facet class is the base class for a localization feature, such as money, time, and number printing. It provides common support for facets and reference management.*

- class [id](#)

*Facet ID class.*

*The ID class provides facets with an index used to identify them. Every facet class must define a public static member [locale::id](#), or be derived from a facet that provides this member, otherwise the facet cannot be used in a locale. The [locale::id](#) ensures that each class type gets a unique identifier.*

### Public Types

- typedef int [category](#)

### Public Member Functions

- [locale](#) () throw ()
- [locale](#) (const [locale](#) &\_\_other) throw ()
- [locale](#) (const [locale](#) &\_\_base, const char \*\_\_s, [category](#) \_\_cat)
- [locale](#) (const [locale](#) &\_\_base, const [locale](#) &\_\_add, [category](#) \_\_cat)
- [locale](#) (const char \*\_\_s)
- template<typename \_Facet >  
  [locale](#) (const [locale](#) &\_\_other, \_Facet \*\_\_f)
- [~locale](#) () throw ()
- template<typename \_Facet >  
  [locale combine](#) (const [locale](#) &\_\_other) const
- [string name](#) () const
- bool [operator!=](#) (const [locale](#) &\_\_other) const throw ()
- template<typename \_Char, typename \_Traits, typename \_Alloc >  
  bool [operator\(\)](#) (const [basic\\_string](#)< \_Char, \_Traits, \_Alloc > &\_\_s1, const [basic\\_string](#)< \_Char, \_Traits, \_Alloc > &\_\_s2) const

- template<typename \_CharT, typename \_Traits, typename \_Alloc >  
bool **operator()** (const [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_s1, const  
[basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_s2) const
- const [locale](#) & **operator=** (const [locale](#) &\_\_other) throw ()
- bool **operator==** (const [locale](#) &\_\_other) const throw ()

### Static Public Member Functions

- static const [locale](#) & **classic** ()
- static [locale](#) **global** (const [locale](#) &)

### Static Public Attributes

- static const [category](#) **none**
- static const [category](#) **ctype**
- static const [category](#) **numeric**
- static const [category](#) **collate**
- static const [category](#) **time**
- static const [category](#) **monetary**
- static const [category](#) **messages**
- static const [category](#) **all**

### Friends

- struct **\_\_use\_cache**
- class **\_Impl**
- class **facet**
- template<typename \_Facet >  
bool **has\_facet** (const [locale](#) &) throw ()
- template<typename \_Facet >  
const \_Facet & **use\_facet** (const [locale](#) &)

#### 5.574.1 Detailed Description

Container class for localization functionality.

The locale class is first a class wrapper for C library locales. It is also an extensible container for user-defined localization. A locale is a collection of facets that implement various localization features such as money, time, and number printing. Constructing C++ locales does not change the C library locale.

This library supports efficient construction and copying of locales through a reference counting implementation of the locale class.

Definition at line 64 of file locale\_classes.h.



## 5.574.2 Member Typedef Documentation

### 5.574.2.1 `typedef int std::locale::category`

Definition of `locale::category`.

Definition at line 69 of file `locale_classes.h`.

## 5.574.3 Constructor & Destructor Documentation

### 5.574.3.1 `std::locale::locale ( ) throw ()`

Default constructor.

Constructs a copy of the global locale. If no locale has been explicitly set, this is the C locale.

Referenced by `combine()`.

### 5.574.3.2 `std::locale::locale ( const locale & __other ) throw ()`

Copy constructor.

Constructs a copy of *other*.

#### Parameters

*other* The locale to copy.

### 5.574.3.3 `std::locale::locale ( const char * __s ) [explicit]`

Named locale constructor.

Constructs a copy of the named C library locale.

#### Parameters

*s* Name of the locale to construct.

#### Exceptions

`std::runtime_error` if *s* is null or an undefined locale.

#### 5.574.3.4 std::locale::locale ( const locale & \_\_base, const char \* \_\_s, category \_\_cat )

Construct locale with facets from another locale.

Constructs a copy of the locale *base*. The facets specified by *cat* are replaced with those from the locale named by *s*. If *base* is named, this locale instance will also be named.

##### Parameters

*base* The locale to copy.

*s* Name of the locale to use facets from.

*cat* Set of categories defining the facets to use from *s*.

##### Exceptions

[\*std::runtime\\_error\*](#) if *s* is null or an undefined locale.

#### 5.574.3.5 std::locale::locale ( const locale & \_\_base, const locale & \_\_add, category \_\_cat )

Construct locale with facets from another locale.

Constructs a copy of the locale *base*. The facets specified by *cat* are replaced with those from the locale *add*. If *base* and *add* are named, this locale instance will also be named.

##### Parameters

*base* The locale to copy.

*add* The locale to use facets from.

*cat* Set of categories defining the facets to use from *add*.

#### 5.574.3.6 template<typename \_Facet > std::locale::locale ( const locale & \_\_other, \_Facet \* \_\_f )

Construct locale with another facet.

Constructs a copy of the locale *other*. The facet is added to , replacing an existing facet of type *Facet* if there is one. If *f* is null, this locale is a copy of *other*.

**Parameters**

*other* The locale to copy.

*f* The facet to add in.

Definition at line 45 of file locale\_classes.tcc.

**5.574.3.7 std::locale::~~locale ( ) throw ()**

Locale destructor.

**5.574.4 Member Function Documentation****5.574.4.1 static const locale& std::locale::classic ( ) [static]**

Return reference to the C locale.

**5.574.4.2 template<typename \_Facet > locale std::locale::combine ( const locale & \_\_other ) const**

Construct locale with another facet.

Constructs and returns a new copy of this locale. Adds or replaces an existing facet of type Facet from the locale *other* into the new locale.

**Parameters**

*Facet* The facet type to copy from other

*other* The locale to copy from.

**Returns**

Newly constructed locale.

**Exceptions**

[\*std::runtime\\_error\*](#) if other has no facet of type Facet.

Definition at line 63 of file locale\_classes.tcc.

References locale().

**5.574.4.3 static locale std::locale::global ( const locale & ) [static]**

Set global locale.

This function sets the global locale to the argument and returns a copy of the previous global locale. If the argument has a name, it will also call std::setlocale(LC\_ALL, loc.name()).

**Parameters**

*locale* The new locale to make global.

**Returns**

Copy of the old global locale.

**5.574.4.4 string std::locale::name ( ) const**

Return locale name.

**Returns**

Locale name or "\*" if unnamed.

**5.574.4.5 bool std::locale::operator!= ( const locale & \_\_other ) const throw () [inline]**

Locale inequality.

**Parameters**

*other* The locale to compare against.

**Returns**

! (\*this == other)

Definition at line 236 of file locale\_classes.h.

References operator==().

**5.574.4.6** `template<typename _Char , typename _Traits , typename _Alloc >  
bool std::locale::operator() ( const basic_string< _Char, _Traits,  
_Alloc > & __s1, const basic_string< _Char, _Traits, _Alloc > &  
__s2 ) const`

Compare two strings according to collate.

Template operator to compare two strings using the compare function of the collate facet in this locale. One use is to provide the locale to the sort function. For example, a vector *v* of strings could be sorted according to locale *loc* by doing:

```
std::sort(v.begin(), v.end(), loc);
```

#### Parameters

*s1* First string to compare.

*s2* Second string to compare.

#### Returns

True if collate<Char> facet compares *s1* < *s2*, else false.

**5.574.4.7** `const locale& std::locale::operator= ( const locale & __other ) throw  
()`

Assignment operator.

Set this locale to be a copy of *other*.

#### Parameters

*other* The locale to copy.

#### Returns

A reference to this locale.

**5.574.4.8** `bool std::locale::operator== ( const locale & __other ) const throw ()`

Locale equality.

**Parameters**

*other* The locale to compare against.

**Returns**

True if *other* and this refer to the same locale instance, are copies, or have the same name. False otherwise.

Referenced by operator!=().

**5.574.5 Friends And Related Function Documentation****5.574.5.1 template<typename \_Facet > bool has\_facet ( const locale & )  
throw () [friend]**

Test for the presence of a facet.

has\_facet tests the locale argument for the presence of the facet type provided as the template parameter. Facets derived from the facet parameter will also return true.

**Parameters**

*Facet* The facet type to test the presence of.

*locale* The locale to test.

**Returns**

true if locale contains a facet of type Facet, else false.

**5.574.5.2 template<typename \_Facet > const \_Facet& use\_facet ( const locale  
& ) [friend]**

Return a facet.

use\_facet looks for and returns a reference to a facet of type Facet where Facet is the template parameter. If has\_facet(locale) is true, there is a suitable facet to return. It throws [std::bad\\_cast](#) if the locale doesn't contain a facet of type Facet.

**Parameters**

*Facet* The facet type to access.

*locale* The locale to use.

**Returns**

Reference to facet of type Facet.

**Exceptions**

[\*std::bad\\_cast\*](#) if locale doesn't contain a facet of type Facet.

**5.574.6 Member Data Documentation****5.574.6.1 const category std::locale::all [static]**

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in locale.cc

Definition at line 107 of file locale\_classes.h.

**5.574.6.2 const category std::locale::collate [static]**

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in locale.cc

Definition at line 103 of file locale\_classes.h.

**5.574.6.3 const category std::locale::ctype [static]**

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in locale.cc

Definition at line 101 of file locale\_classes.h.

**5.574.6.4 const category std::locale::messages [static]**

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match \_S\_facet\_categories definition in locale.cc

Definition at line 106 of file locale\_classes.h.

**5.574.6.5 const category std::locale::monetary [static]**

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match \_S\_facet\_categories definition in locale.cc

Definition at line 105 of file locale\_classes.h.

**5.574.6.6 const category std::locale::none [static]**

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match \_S\_facet\_categories definition in locale.cc

Definition at line 100 of file locale\_classes.h.

**5.574.6.7 const category std::locale::numeric [static]**

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.



NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 102 of file `locale_classes.h`.

#### 5.574.6.8 `const category std::locale::time` `[static]`

Category values.

The standard category values are `none`, `ctype`, `numeric`, `collate`, `time`, `monetary`, and `messages`. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 104 of file `locale_classes.h`.

The documentation for this class was generated from the following files:

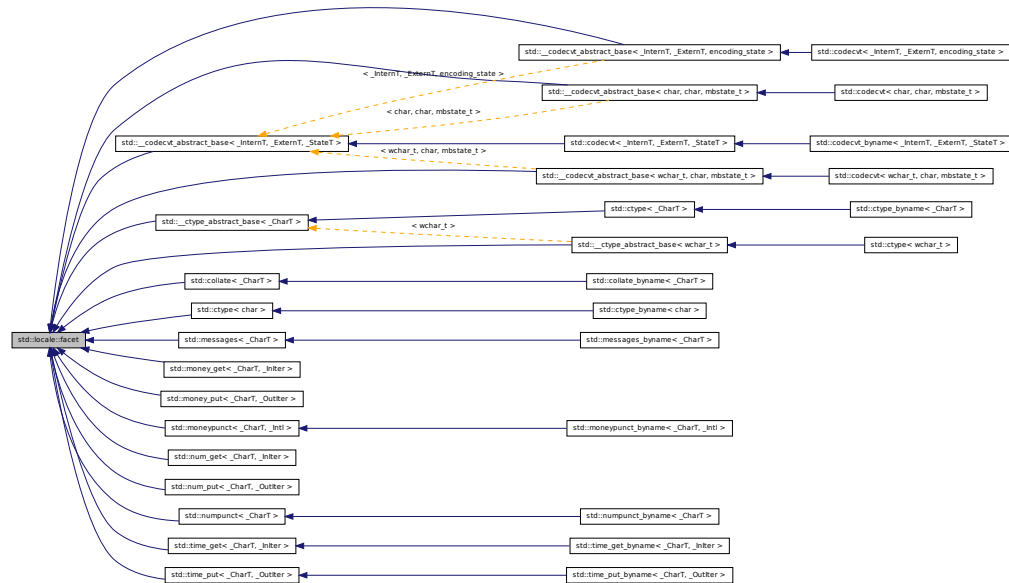
- [locale\\_classes.h](#)
- [locale\\_classes.tcc](#)

### 5.575 `std::locale::facet` Class Reference

Localization functionality base class.

The facet class is the base class for a localization feature, such as money, time, and number printing. It provides common support for facets and reference management.

Inheritance diagram for std::locale::facet:



### Protected Member Functions

- [facet](#) (size\_t \_\_refs=0) throw ()
- virtual [~facet](#) ()

### Static Protected Member Functions

- static \_\_c\_locale [\\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale [\\_S\\_get\\_c\\_locale](#) ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static \_\_c\_locale [\\_S\\_lc\\_ctype\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \*\_\_s)

### Friends

- class [locale](#)
- class [locale::\\_Impl](#)

### 5.575.1 Detailed Description

Localization functionality base class.

The facet class is the base class for a localization feature, such as money, time, and number printing. It provides common support for facets and reference management. Facets may not be copied or assigned.

Definition at line 339 of file `locale_classes.h`.

### 5.575.2 Constructor & Destructor Documentation

#### 5.575.2.1 `std::locale::facet::facet ( size_t __refs = 0 ) throw () [inline, explicit, protected]`

Facet constructor.

This is the constructor provided by the standard. If `refs` is 0, the facet is destroyed when the last referencing locale is destroyed. Otherwise the facet will never be destroyed.

#### Parameters

*refs* The initial value for reference count.

Definition at line 371 of file `locale_classes.h`.

#### 5.575.2.2 `virtual std::locale::facet::~~facet ( ) [protected, virtual]`

Facet destructor.

The documentation for this class was generated from the following file:

- [locale\\_classes.h](#)

## 5.576 `std::locale::id` Class Reference

Facet ID class.

The ID class provides facets with an index used to identify them. Every facet class must define a public static member `locale::id`, or be derived from a facet that provides this member, otherwise the facet cannot be used in a locale. The `locale::id` ensures that each class type gets a unique identifier.

## Public Member Functions

- [id](#) ()
- `size_t _M_id` () const throw ()

## Friends

- `template<typename _Facet >`  
`bool has\_facet (const locale &) throw ()`
- class **locale**
- class **locale::\_Impl**
- `template<typename _Facet >`  
`const _Facet & use\_facet (const locale &)`

### 5.576.1 Detailed Description

Facet ID class.

The ID class provides facets with an index used to identify them. Every facet class must define a public static member [locale::id](#), or be derived from a facet that provides this member, otherwise the facet cannot be used in a locale. The [locale::id](#) ensures that each class type gets a unique identifier.

Definition at line 437 of file locale\_classes.h.

### 5.576.2 Constructor & Destructor Documentation

#### 5.576.2.1 std::locale::id::id ( ) [inline]

Constructor.

Definition at line 468 of file locale\_classes.h.

### 5.576.3 Friends And Related Function Documentation

#### 5.576.3.1 `template<typename _Facet > bool has_facet ( const locale & )` `throw () [friend]`

Test for the presence of a facet.

`has_facet` tests the locale argument for the presence of the facet type provided as the template parameter. Facets derived from the facet parameter will also return true.

**Parameters**

*Facet* The facet type to test the presence of.

*locale* The locale to test.

**Returns**

true if locale contains a facet of type Facet, else false.

**5.576.3.2 `template<typename _Facet> const _Facet& use_facet ( const locale & ) [friend]`**

Return a facet.

`use_facet` looks for and returns a reference to a facet of type Facet where Facet is the template parameter. If `has_facet(locale)` is true, there is a suitable facet to return. It throws [std::bad\\_cast](#) if the locale doesn't contain a facet of type Facet.

**Parameters**

*Facet* The facet type to access.

*locale* The locale to use.

**Returns**

Reference to facet of type Facet.

**Exceptions**

[std::bad\\_cast](#) if locale doesn't contain a facet of type Facet.

The documentation for this class was generated from the following file:

- [locale\\_classes.h](#)

**5.577 `std::lock_guard<_Mutex>` Class Template Reference**

Scoped lock idiom.

**Public Types**

- `typedef _Mutex mutex_type`

### Public Member Functions

- `lock_guard` (`mutex_type &__m`)
- `lock_guard` (`mutex_type &__m`, `adopt_lock_t`)
- `lock_guard` (`const lock_guard &`)
- `lock_guard & operator=` (`const lock_guard &`)

#### 5.577.1 Detailed Description

`template<typename _Mutex> class std::lock_guard< _Mutex >`

Scoped lock idiom.

Definition at line 439 of file `mutex`.

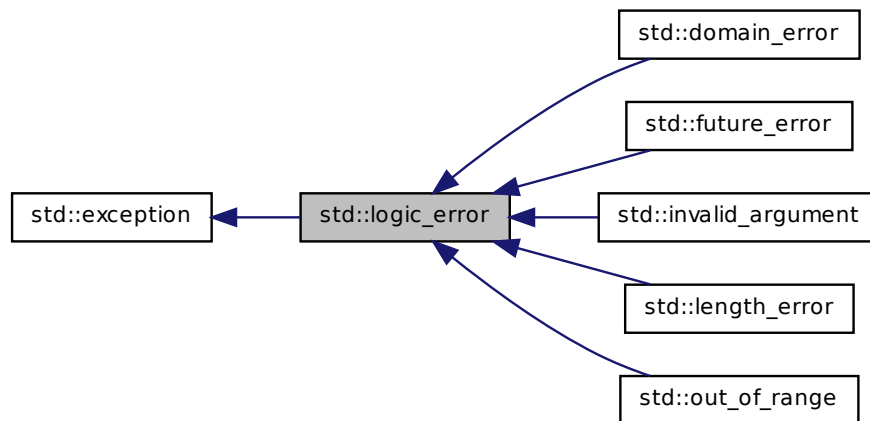
The documentation for this class was generated from the following file:

- `mutex`

## 5.578 `std::logic_error` Class Reference

One of two subclasses of exception.

Inheritance diagram for `std::logic_error`:



## Public Member Functions

- `logic_error` (const `string` &\_\_arg)
- virtual const char \* `what` () const throw ()

### 5.578.1 Detailed Description

One of two subclasses of exception. Logic errors represent problems in the internal logic of a program; in theory, these are preventable, and even detectable before the program runs (e.g., violations of class invariants).

Definition at line 56 of file `stdexcept`.

### 5.578.2 Constructor & Destructor Documentation

#### 5.578.2.1 `std::logic_error::logic_error ( const string & __arg ) [explicit]`

Takes a character string describing the error.

### 5.578.3 Member Function Documentation

#### 5.578.3.1 `virtual const char* std::logic_error::what ( ) const throw () [virtual]`

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from `std::exception`.

Reimplemented in `std::future_error`.

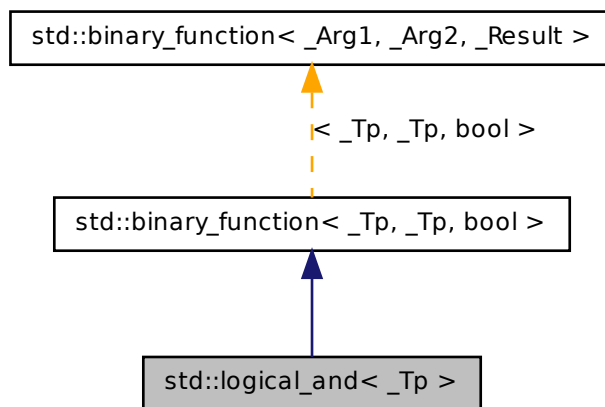
The documentation for this class was generated from the following file:

- `stdexcept`

## 5.579 `std::logical_and< _Tp >` Struct Template Reference

One of the [Boolean operations functors](#).

Inheritance diagram for std::logical\_and< \_Tp >:



### Public Types

- typedef `_Tp` `first_argument_type`
- typedef `bool` `result_type`
- typedef `_Tp` `second_argument_type`

### Public Member Functions

- `bool operator() (const _Tp &__x, const _Tp &__y) const`

#### 5.579.1 Detailed Description

**template<typename \_Tp> struct std::logical\_and< \_Tp >**

One of the [Boolean operations functors](#).

Definition at line 269 of file `stl_function.h`.



### 5.579.2 Member Typedef Documentation

**5.579.2.1** `typedef _Tp std::binary_function<_Tp, _Tp, bool  
>::first_argument_type [inherited]`

`first_argument_type` is the type of the first argument

Definition at line 118 of file `stl_function.h`.

**5.579.2.2** `typedef bool std::binary_function<_Tp, _Tp, bool>::result_type  
[inherited]`

`result_type` is the return type

Definition at line 124 of file `stl_function.h`.

**5.579.2.3** `typedef _Tp std::binary_function<_Tp, _Tp, bool  
>::second_argument_type [inherited]`

`second_argument_type` is the type of the second argument

Definition at line 121 of file `stl_function.h`.

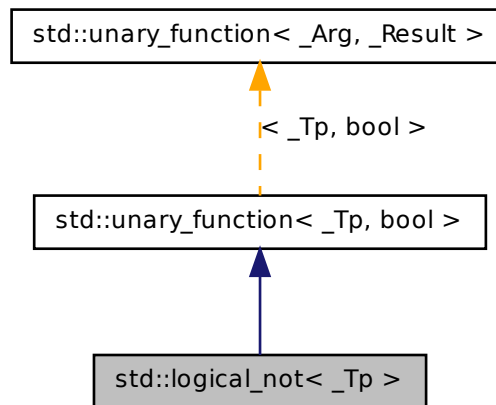
The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.580 `std::logical_not<_Tp>` Struct Template Reference

One of the [Boolean operations functors](#).

Inheritance diagram for `std::logical_not< _Tp >`:



### Public Types

- typedef `_Tp` [argument\\_type](#)
- typedef `bool` [result\\_type](#)

### Public Member Functions

- `bool operator() (const _Tp &__x) const`

#### 5.580.1 Detailed Description

`template<typename _Tp> struct std::logical_not< _Tp >`

One of the [Boolean operations functors](#).

Definition at line 287 of file `stl_function.h`.

### 5.580.2 Member Typedef Documentation

#### 5.580.2.1 `typedef _Tp std::unary_function< _Tp , bool >::argument_type` `[inherited]`

`argument_type` is the type of the argument

Definition at line 105 of file `stl_function.h`.

#### 5.580.2.2 `typedef bool std::unary_function< _Tp , bool >::result_type` `[inherited]`

`result_type` is the return type

Definition at line 108 of file `stl_function.h`.

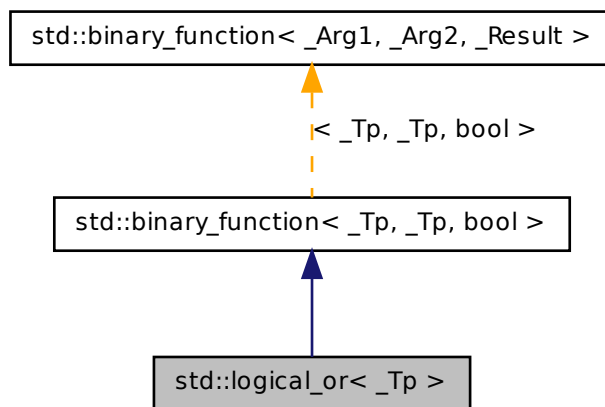
The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

### 5.581 `std::logical_or< _Tp >` Struct Template Reference

One of the [Boolean operations functors](#).

Inheritance diagram for std::logical\_or< \_Tp >:



### Public Types

- typedef `_Tp` `first_argument_type`
- typedef `bool` `result_type`
- typedef `_Tp` `second_argument_type`

### Public Member Functions

- `bool operator() (const _Tp &__x, const _Tp &__y) const`

#### 5.581.1 Detailed Description

**template<typename \_Tp> struct std::logical\_or< \_Tp >**

One of the [Boolean operations functors](#).

Definition at line 278 of file `stl_function.h`.

## 5.582 `std::lognormal_distribution<_RealType>` Class Template Reference 2882

### 5.581.2 Member Typedef Documentation

**5.581.2.1** `typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type [inherited]`

`first_argument_type` is the type of the first argument

Definition at line 118 of file `stl_function.h`.

**5.581.2.2** `typedef bool std::binary_function< _Tp , _Tp , bool >::result_type [inherited]`

`result_type` is the return type

Definition at line 124 of file `stl_function.h`.

**5.581.2.3** `typedef _Tp std::binary_function< _Tp , _Tp , bool >::second_argument_type [inherited]`

`second_argument_type` is the type of the second argument

Definition at line 121 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.582 `std::lognormal_distribution<_RealType>` Class Template Reference

A [lognormal\\_distribution](#) random number distribution.

### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_RealType` [result\\_type](#)

### Public Member Functions

- **lognormal\_distribution** (`_RealType __m=_RealType(0), _RealType __s=_RealType(1)`)
- **lognormal\_distribution** (const `param_type` &\_\_p)
- `_RealType m` () const
- `result_type max` () const
- `result_type min` () const
- `template<typename _UniformRandomNumberGenerator>`  
`result_type operator()` (`_UniformRandomNumberGenerator &__urng`, const `param_type` &\_\_p)
- `template<typename _UniformRandomNumberGenerator>`  
`result_type operator()` (`_UniformRandomNumberGenerator &__urng`)
- void `param` (const `param_type` &\_\_param)
- `param_type param` () const
- void `reset` ()
- `_RealType s` () const

### Friends

- `template<typename _RealType1, typename _CharT, typename _Traits>`  
`std::basic_ostream<_CharT, _Traits> & operator<<` (`std::basic_ostream<_CharT, _Traits> &`, const `std::lognormal_distribution<_RealType1>` &)
- `template<typename _RealType1>`  
`bool operator==` (const `std::lognormal_distribution<_RealType1>` &\_\_d1, const `std::lognormal_distribution<_RealType1>` &\_\_d2)
- `template<typename _RealType1, typename _CharT, typename _Traits>`  
`std::basic_istream<_CharT, _Traits> & operator>>` (`std::basic_istream<_CharT, _Traits> &`, `std::lognormal_distribution<_RealType1>` &)

#### 5.582.1 Detailed Description

`template<typename _RealType = double> class std::lognormal_distribution<_RealType>`

A `lognormal_distribution` random number distribution. The formula for the normal probability mass function is

$$p(x|m, s) = \frac{1}{sx\sqrt{2\pi}} \exp -\frac{(\ln x - m)^2}{2s^2}$$

Definition at line 2151 of file random.h.

## 5.582.2 Member Typedef Documentation

**5.582.2.1** `template<typename _RealType = double> typedef _RealType  
std::lognormal_distribution<_RealType>::result_type`

The type of the range of the distribution.

Definition at line 2158 of file random.h.

## 5.582.3 Member Function Documentation

**5.582.3.1** `template<typename _RealType = double> result_type  
std::lognormal_distribution<_RealType>::max ( ) const  
[inline]`

Returns the least upper bound value of the distribution.

Definition at line 2242 of file random.h.

**5.582.3.2** `template<typename _RealType = double> result_type  
std::lognormal_distribution<_RealType>::min ( ) const  
[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 2235 of file random.h.

**5.582.3.3** `template<typename _RealType = double> template<typename  
_UniformRandomNumberGenerator> result_type  
std::lognormal_distribution<_RealType>::operator() (   
_UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 2250 of file random.h.

References `std::lognormal_distribution<_RealType>::operator()()`, and  
`std::lognormal_distribution<_RealType>::param()`.

Referenced by `std::lognormal_distribution<_RealType>::operator()()`.

**5.582.3.4** `template<typename _RealType = double> void  
std::lognormal_distribution<_RealType>::param ( const  
param_type & __param ) [inline]`

Sets the parameter set of the distribution.

#### Parameters

`__param` The new parameter set of the distribution.

Definition at line 2228 of file `random.h`.

**5.582.3.5** `template<typename _RealType = double> param_type  
std::lognormal_distribution<_RealType>::param ( ) const  
[inline]`

Returns the parameter set of the distribution.

Definition at line 2220 of file `random.h`.

Referenced by `std::lognormal_distribution<_RealType>::operator()()`.

**5.582.3.6** `template<typename _RealType = double> void  
std::lognormal_distribution<_RealType>::reset ( ) [inline]`

Resets the distribution state.

Definition at line 2202 of file `random.h`.

References `std::normal_distribution<_RealType>::reset()`.

#### 5.582.4 Friends And Related Function Documentation

**5.582.4.1** `template<typename _RealType = double> template<typename  
_RealType1 , typename _CharT , typename _Traits >  
std::basic_ostream<_CharT, _Traits>& operator<<  
( std::basic_ostream<_CharT, _Traits> & , const  
std::lognormal_distribution<_RealType1> & ) [friend]`

Inserts a `lognormal_distribution` random number distribution `__x` into the output stream `__os`.



## 5.582 `std::lognormal_distribution<_RealType>` Class Template Reference 2886

### Parameters

- `__os` An output stream.
- `__x` A `lognormal_distribution` random number distribution.

### Returns

The output stream with the state of `__x` inserted or in an error state.

**5.582.4.2** `template<typename _RealType = double> template<typename _RealType1 > bool operator==( const std::lognormal_distribution<_RealType1 > & __d1, const std::lognormal_distribution<_RealType1 > & __d2 ) [friend]`

Return true if two lognormal distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 2266 of file `random.h`.

**5.582.4.3** `template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename _Traits > std::basic_istream<_CharT, _Traits>& operator>>( std::basic_istream<_CharT, _Traits> & , std::lognormal_distribution<_RealType1> & ) [friend]`

Extracts a `lognormal_distribution` random number distribution `__x` from the input stream `__is`.

### Parameters

- `__is` An input stream.
- `__x` A `lognormal_distribution` random number generator engine.

### Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

## 5.583 `std::lognormal_distribution< _RealType >::param_type` Struct Reference

### Public Types

- typedef [lognormal\\_distribution](#)< \_RealType > **distribution\_type**

### Public Member Functions

- **param\_type** (\_RealType \_\_m=\_RealType(0), \_RealType \_\_s=\_RealType(1))
- \_RealType **m** () const
- \_RealType **s** () const

### Friends

- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

#### 5.583.1 Detailed Description

`template<typename _RealType = double> struct std::lognormal_distribution< _RealType >::param_type`

Parameter type.

Definition at line 2160 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.584 `std::make_signed< _Tp >` Struct Template Reference

[make\\_signed](#)

### Public Types

- typedef `__make_signed_selector< _Tp >::__type` **type**

#### 5.584.1 Detailed Description

`template<typename _Tp> struct std::make_signed< _Tp >`

[make\\_signed](#)

Definition at line 1096 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.585 `std::make_unsigned<_Tp>` Struct Template Reference

[make\\_unsigned](#)

### Public Types

- typedef `__make_unsigned_selector<_Tp>::__type` **type**

#### 5.585.1 Detailed Description

`template<typename _Tp> struct std::make_unsigned<_Tp>`

[make\\_unsigned](#)

Definition at line 1019 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.586 `std::map<_Key, _Tp, _Compare, _Alloc>` Class Template Reference

A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.

### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_Rep_type::const_iterator` **const\_iterator**
- typedef `_Pair_alloc_type::const_pointer` **const\_pointer**
- typedef `_Pair_alloc_type::const_reference` **const\_reference**
- typedef `_Rep_type::const_reverse_iterator` **const\_reverse\_iterator**
- typedef `_Rep_type::difference_type` **difference\_type**
- typedef `_Rep_type::iterator` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**

- typedef `_Tp mapped_type`
- typedef `_Pair_alloc_type::pointer pointer`
- typedef `_Pair_alloc_type::reference reference`
- typedef `_Rep_type::reverse_iterator reverse_iterator`
- typedef `_Rep_type::size_type size_type`
- typedef `std::pair< const _Key, _Tp > value_type`

### Public Member Functions

- `map ()`
- `map (const _Compare &__comp, const allocator_type &__a=allocator_type())`
- `map (map &&__x)`
- `map (initializer_list< value_type > __l, const _Compare &__c=_Compare(), const allocator_type &__a=allocator_type())`
- `map (const map &__x)`
- `template<typename _InputIterator >`  
`map (_InputIterator __first, _InputIterator __last)`
- `template<typename _InputIterator >`  
`map (_InputIterator __first, _InputIterator __last, const _Compare &__comp, const allocator_type &__a=allocator_type())`
- `mapped_type & at (const key_type &__k)`
- `const mapped_type & at (const key_type &__k) const`
- `iterator begin ()`
- `const_iterator begin () const`
- `const_iterator cbegin () const`
- `const_iterator end () const`
- `void clear ()`
- `size_type count (const key_type &__x) const`
- `const_reverse_iterator crbegin () const`
- `const_reverse_iterator crend () const`
- `bool empty () const`
- `iterator end ()`
- `const_iterator end () const`
- `std::pair< iterator, iterator > equal_range (const key_type &__x)`
- `std::pair< const_iterator, const_iterator > equal_range (const key_type &__x) const`
- `iterator erase (const_iterator __position)`
- `size_type erase (const key_type &__x)`
- `iterator erase (const_iterator __first, const_iterator __last)`
- `iterator find (const key_type &__x)`
- `const_iterator find (const key_type &__x) const`
- `allocator_type get_allocator () const`

- `template<typename _Pair, typename = typename std::enable_if<std::is_convertible<_Pair, value_type>::value>::type>`  
`std::pair< iterator, bool > insert (_Pair &&__x)`
- `void insert (std::initializer_list< value_type > __list)`
- `iterator insert (const_iterator __position, const value_type &__x)`
- `template<typename _InputIterator >`  
`void insert (_InputIterator __first, _InputIterator __last)`
- `template<typename _Pair, typename = typename std::enable_if<std::is_convertible<_Pair, value_type>::value>::type>`  
`iterator insert (const_iterator __position, _Pair &&__x)`
- `std::pair< iterator, bool > insert (const value_type &__x)`
- `key_compare key_comp () const`
- `iterator lower_bound (const key_type &__x)`
- `const_iterator lower_bound (const key_type &__x) const`
- `size_type max_size () const`
- `map & operator= (initializer_list< value_type > __l)`
- `map & operator= (map &&__x)`
- `map & operator= (const map &__x)`
- `mapped_type & operator[] (const key_type &__k)`
- `mapped_type & operator[] (key_type &&__k)`
- `const_reverse_iterator rbegin () const`
- `reverse_iterator rbegin ()`
- `const_reverse_iterator rend () const`
- `reverse_iterator rend ()`
- `size_type size () const`
- `void swap (map &__x)`
- `iterator upper_bound (const key_type &__x)`
- `const_iterator upper_bound (const key_type &__x) const`
- `value_compare value_comp () const`

## Friends

- `template<typename _K1, typename _T1, typename _C1, typename _A1 >`  
`bool operator< (const map< _K1, _T1, _C1, _A1 > &, const map< _K1, _T1, _C1, _A1 > &)`
- `template<typename _K1, typename _T1, typename _C1, typename _A1 >`  
`bool operator== (const map< _K1, _T1, _C1, _A1 > &, const map< _K1, _T1, _C1, _A1 > &)`

### 5.586.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> class
std::map< _Key, _Tp, _Compare, _Alloc >
```

A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time. Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using unique keys). For a `map<Key, T>` the `key_type` is `Key`, the `mapped_type` is `T`, and the `value_type` is `std::pair<const Key, T>`.

Maps support bidirectional iterators.

The private tree data is declared exactly the same way for `map` and `multimap`; the distinction is made entirely in how the tree functions are called (`*_unique` versus `*_equal`, same as the standard).

Definition at line 88 of file `stl_map.h`.

### 5.586.2 Constructor & Destructor Documentation

```
5.586.2.1 template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> std::map< _Key, _Tp, _Compare, _Alloc >::map (
) [inline]
```

Default constructor creates no elements.

Definition at line 152 of file `stl_map.h`.

```
5.586.2.2 template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> std::map< _Key, _Tp, _Compare, _Alloc >::map
(const _Compare & __comp, const allocator_type & __a =
allocator_type()) [inline, explicit]
```

Creates a map with no elements.

#### Parameters

*comp* A comparison object.

*a* An allocator object.

## 5.586 std::map< \_Key, \_Tp, \_Compare, \_Alloc > Class Template Reference 2892

---

Definition at line 161 of file stl\_map.h.

**5.586.2.3** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> std::map< _Key, _Tp, _Compare, _Alloc >::map (  
const map< _Key, _Tp, _Compare, _Alloc > & __x ) [inline]`

Map copy constructor.

### Parameters

*x* A map of identical element and allocator types.

The newly-created map uses a copy of the allocation object used by *x*.

Definition at line 172 of file stl\_map.h.

**5.586.2.4** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> std::map< _Key, _Tp, _Compare, _Alloc >::map (  
map< _Key, _Tp, _Compare, _Alloc > && __x ) [inline]`

Map move constructor.

### Parameters

*x* A map of identical element and allocator types.

The newly-created map contains the exact contents of *x*. The contents of *x* are a valid, but unspecified map.

Definition at line 183 of file stl\_map.h.

**5.586.2.5** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> std::map< _Key, _Tp, _Compare, _Alloc >::map (  
initializer_list< value_type > __l, const _Compare & __c =  
_Compare(), const allocator_type & __a = allocator_type()  
) [inline]`

Builds a map from an [initializer\\_list](#).

**Parameters**

- l* An [initializer\\_list](#).
- comp* A comparison object.
- a* An allocator object.

Create a map consisting of copies of the elements in the [initializer\\_list](#) *l*. This is linear in N if the range is already sorted, and NlogN otherwise (where N is *l.size()*).

Definition at line 197 of file `stl_map.h`.

```
5.586.2.6 template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> template<typename _InputIterator> std::map<
_Key, _Tp, _Compare, _Alloc>::map (_InputIterator __first,
_InputIterator __last) [inline]
```

Builds a map from a range.

**Parameters**

- first* An input iterator.
- last* An input iterator.

Create a map consisting of copies of the elements from [first,last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(first,last)).

Definition at line 214 of file `stl_map.h`.

```
5.586.2.7 template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> template<typename _InputIterator> std::map<
_Key, _Tp, _Compare, _Alloc>::map (_InputIterator __first,
_InputIterator __last, const _Compare & __comp, const
allocator_type & __a = allocator_type()) [inline]
```

Builds a map from a range.

**Parameters**

- first* An input iterator.
- last* An input iterator.



## 5.586 std::map< \_Key, \_Tp, \_Compare, \_Alloc > Class Template Reference 2894

---

*comp* A comparison functor.

*a* An allocator object.

Create a map consisting of copies of the elements from [first,last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(first,last)).

Definition at line 230 of file stl\_map.h.

### 5.586.3 Member Function Documentation

**5.586.3.1** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> mapped_type& std::map<_Key, _Tp, _Compare, _Alloc>::at( const key_type & __k ) [inline]`

Access to map data.

#### Parameters

*k* The key for which data should be retrieved.

#### Returns

A reference to the data whose key is equivalent to *k*, if such a data is present in the map.

#### Exceptions

*std::out\_of\_range* If no such data is present.

Definition at line 482 of file stl\_map.h.

References `std::map<_Key, _Tp, _Compare, _Alloc>::end()`, `std::map<_Key, _Tp, _Compare, _Alloc>::key_comp()`, and `std::map<_Key, _Tp, _Compare, _Alloc>::lower_bound()`.

**5.586.3.2** `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> iterator std::map<_Key, _Tp, _Compare, _Alloc>::begin( ) [inline]`

Returns a read/write iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 309 of file stl\_map.h.

**5.586.3.3** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> const_iterator std::map< _Key, _Tp, _Compare,  
_Alloc >::begin ( ) const [inline]`

Returns a read-only (constant) iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 318 of file stl\_map.h.

**5.586.3.4** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> const_iterator std::map< _Key, _Tp, _Compare,  
_Alloc >::cbegin ( ) const [inline]`

Returns a read-only (constant) iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 382 of file stl\_map.h.

**5.586.3.5** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> const_iterator std::map< _Key, _Tp, _Compare,  
_Alloc >::cend ( ) const [inline]`

Returns a read-only (constant) iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 391 of file stl\_map.h.

**5.586.3.6** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> void std::map< _Key, _Tp, _Compare, _Alloc  
>::clear ( ) [inline]`

Erases all elements in a map. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 703 of file stl\_map.h.

Referenced by std::map< \_Key, \_Tp, \_Compare, \_Alloc >::operator=().

**5.586.3.7** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> size_type std::map< _Key, _Tp, _Compare, _Alloc  
>::count ( const key_type & __x ) const [inline]`

Finds the number of elements with given key.

#### Parameters

*x* Key of (key, value) pairs to be located.

#### Returns

Number of elements with specified key.

This function only makes sense for multimaps; for map the result will either be 0 (not present) or 1 (present).

Definition at line 763 of file stl\_map.h.

**5.586.3.8** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> const_reverse_iterator std::map< _Key, _Tp,  
_Compare, _Alloc >::crbegin ( ) const [inline]`

Returns a read-only (constant) reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

Definition at line 400 of file stl\_map.h.

**5.586.3.9** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> const_reverse_iterator std::map< _Key, _Tp,  
_Compare, _Alloc >::crend ( ) const [inline]`

Returns a read-only (constant) reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

Definition at line 409 of file stl\_map.h.

**5.586.3.10** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> bool std::map< _Key, _Tp, _Compare, _Alloc  
>::empty ( ) const [inline]`

Returns true if the map is empty. (Thus `begin()` would equal `end()`.)

Definition at line 418 of file `stl_map.h`.

**5.586.3.11** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> const_iterator std::map< _Key, _Tp, _Compare,  
_Alloc >::end ( ) const [inline]`

Returns a read-only (constant) iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 336 of file `stl_map.h`.

**5.586.3.12** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> iterator std::map< _Key, _Tp, _Compare, _Alloc  
>::end ( ) [inline]`

Returns a read/write iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 327 of file `stl_map.h`.

Referenced by `std::map< _Key, _Tp, _Compare, _Alloc >::at()`, and `std::map< _Key, _Tp, _Compare, _Alloc >::operator[]()`.

**5.586.3.13** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> std::pair<const_iterator, const_iterator>  
std::map< _Key, _Tp, _Compare, _Alloc >::equal_range ( const  
key_type & __x ) const [inline]`

Finds a subsequence matching given key.

#### Parameters

*x* Key of (key, value) pairs to be located.

## 5.586 `std::map<_Key, _Tp, _Compare, _Alloc>` Class Template Reference 2898

### Returns

Pair of read-only (constant) iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
 c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multimaps.

Definition at line 851 of file `stl_map.h`.

```
5.586.3.14 template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> std::pair<iterator, iterator> std::map<_Key,
_Tp, _Compare, _Alloc>::equal_range (const key_type & __x)
[inline]
```

Finds a subsequence matching given key.

### Parameters

*x* Key of (key, value) pairs to be located.

### Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
 c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multimaps.

Definition at line 832 of file `stl_map.h`.

**5.586.3.15** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> iterator std::map< _Key, _Tp, _Compare,  
_Alloc >::erase ( const_iterator __first, const_iterator __last )  
[inline]`

Erases a [first,last) range of elements from a map.

#### Parameters

*first* Iterator pointing to the start of the range to be erased.

*last* Iterator pointing to the end of the range to be erased.

#### Returns

The iterator *last*.

This function erases a sequence of elements from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 662 of file stl\_map.h.

**5.586.3.16** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> iterator std::map< _Key, _Tp, _Compare, _Alloc  
>::erase ( const_iterator __position ) [inline]`

Erases an element from a map.

#### Parameters

*position* An iterator pointing to the element to be erased.

#### Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 613 of file stl\_map.h.

**5.586.3.17** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> size_type std::map< _Key, _Tp, _Compare, _Alloc  
>::erase ( const key_type & __x ) [inline]`

Erases elements according to the provided key.

#### Parameters

*x* Key of element to be erased.

#### Returns

The number of elements erased.

This function erases all the elements located by the given key from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 643 of file `stl_map.h`.

**5.586.3.18** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> iterator std::map< _Key, _Tp, _Compare, _Alloc  
>::find ( const key_type & __x ) [inline]`

Tries to locate an element in a map.

#### Parameters

*x* Key of (key, value) pair to be located.

#### Returns

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 736 of file `stl_map.h`.

**5.586.3.19** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> const_iterator std::map< _Key, _Tp, _Compare,  
_Alloc >::find ( const key_type & __x ) const [inline]`

Tries to locate an element in a map.

#### Parameters

*x* Key of (key, value) pair to be located.

#### Returns

Read-only (constant) iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns a constant iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 751 of file `stl_map.h`.

**5.586.3.20** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> allocator_type std::map< _Key, _Tp, _Compare,  
_Alloc >::get_allocator ( ) const [inline]`

Get a copy of the memory allocation object.

Definition at line 299 of file `stl_map.h`.

**5.586.3.21** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> iterator std::map< _Key, _Tp, _Compare, _Alloc  
>::insert ( const_iterator __position, const value_type & __x )  
[inline]`

Attempts to insert a `std::pair` into the map.

#### Parameters

*position* An iterator that serves as a hint as to where the pair should be inserted.



*x* Pair to be inserted (see [std::make\\_pair](#) for easy creation of pairs).

### Returns

An iterator that points to the element with key of *x* (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument [insert\(\)](#) does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html> for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 567 of file stl\_map.h.

```
5.586.3.22 template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> std::pair<iterator, bool> std::map<_Key,
_Tp, _Compare, _Alloc >::insert (const value_type & __x)
[inline]
```

Attempts to insert a [std::pair](#) into the map.

### Parameters

*x* Pair to be inserted (see [std::make\\_pair](#) for easy creation of pairs).

### Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map.

Insertion requires logarithmic time.

Definition at line 517 of file stl\_map.h.

Referenced by `std::map< _Key, _Tp, _Compare, _Alloc >::operator=()`, and `std::map< _Key, _Tp, _Compare, _Alloc >::operator[]()`.

**5.586.3.23** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> void std::map< _Key, _Tp, _Compare, _Alloc  
>::insert ( std::initializer_list< value_type > __list ) [inline]`

Attempts to insert a list of std::pairs into the map.

#### Parameters

*list* A std::initializer\_list<value\_type> of pairs to be inserted.

Complexity similar to that of the range constructor.

Definition at line 538 of file stl\_map.h.

References std::map< \_Key, \_Tp, \_Compare, \_Alloc >::insert().

Referenced by std::map< \_Key, \_Tp, \_Compare, \_Alloc >::insert().

**5.586.3.24** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> template<typename _InputIterator > void  
std::map< _Key, _Tp, _Compare, _Alloc >::insert ( _InputIterator  
__first, _InputIterator __last ) [inline]`

Template function that attempts to insert a range of elements.

#### Parameters

*first* Iterator pointing to the start of the range to be inserted.

*last* Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 593 of file stl\_map.h.

**5.586.3.25** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> key_compare std::map< _Key, _Tp, _Compare,  
_Alloc >::key_comp ( ) const [inline]`

Returns the key comparison object out of which the map was constructed.

Definition at line 712 of file stl\_map.h.

## 5.586 std::map< \_Key, \_Tp, \_Compare, \_Alloc > Class Template Reference 2904

Referenced by std::map< \_Key, \_Tp, \_Compare, \_Alloc >::at(), and std::map< \_Key, \_Tp, \_Compare, \_Alloc >::operator[]().

**5.586.3.26** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> iterator std::map< _Key, _Tp, _Compare, _Alloc  
>::lower_bound ( const key_type & __x ) [inline]`

Finds the beginning of a subsequence matching given key.

### Parameters

*x* Key of (key, value) pair to be located.

### Returns

Iterator pointing to first element equal to or greater than key, or `end()`.

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or `end()` if no such element exists.

Definition at line 778 of file stl\_map.h.

Referenced by std::map< \_Key, \_Tp, \_Compare, \_Alloc >::at(), and std::map< \_Key, \_Tp, \_Compare, \_Alloc >::operator[]().

**5.586.3.27** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> const_iterator std::map< _Key, _Tp, _Compare,  
_Alloc >::lower_bound ( const key_type & __x ) const [inline]`

Finds the beginning of a subsequence matching given key.

### Parameters

*x* Key of (key, value) pair to be located.

### Returns

Read-only (constant) iterator pointing to first element equal to or greater than key, or `end()`.

## 5.586 std::map< \_Key, \_Tp, \_Compare, \_Alloc > Class Template Reference 2905

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or [end\(\)](#) if no such element exists.

Definition at line 793 of file `stl_map.h`.

**5.586.3.28** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> size_type std::map< _Key, _Tp, _Compare, _Alloc  
>::max_size ( ) const [inline]`

Returns the maximum size of the map.

Definition at line 428 of file `stl_map.h`.

**5.586.3.29** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> map& std::map< _Key, _Tp, _Compare, _Alloc  
>::operator= ( initializer_list< value_type > __l ) [inline]`

Map list assignment operator.

### Parameters

*l* An [initializer\\_list](#).

This function fills a map with copies of the elements in the initializer list *l*.

Note that the assignment completely changes the map and that the resulting map's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 289 of file `stl_map.h`.

References `std::map< _Key, _Tp, _Compare, _Alloc >::clear()`, and `std::map< _Key, _Tp, _Compare, _Alloc >::insert()`.

**5.586.3.30** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> map& std::map< _Key, _Tp, _Compare, _Alloc  
>::operator= ( map< _Key, _Tp, _Compare, _Alloc > && __x )  
[inline]`

Map move assignment operator.

## 5.586 std::map< \_Key, \_Tp, \_Compare, \_Alloc > Class Template Reference 2906

---

### Parameters

*x* A map of identical element and allocator types.

The contents of *x* are moved into this map (without copying). *x* is a valid, but unspecified map.

Definition at line 268 of file stl\_map.h.

References std::map< \_Key, \_Tp, \_Compare, \_Alloc >::clear(), and std::map< \_Key, \_Tp, \_Compare, \_Alloc >::swap().

```
5.586.3.31 template<typename _Key, typename _Tp, typename _Compare =
 std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
 _Key, _Tp>>> map& std::map< _Key, _Tp, _Compare, _Alloc
 >::operator= (const map< _Key, _Tp, _Compare, _Alloc > & __x
) [inline]
```

Map assignment operator.

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

### Parameters

*x* A map of identical element and allocator types.

All the elements of *x* are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 253 of file stl\_map.h.

```
5.586.3.32 template<typename _Key, typename _Tp, typename _Compare =
 std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
 _Key, _Tp>>> mapped_type& std::map< _Key, _Tp, _Compare,
 _Alloc >::operator[] (const key_type & __k) [inline]
```

Subscript ( [] ) access to map data.

### Parameters

*k* The key for which data should be retrieved.

### Returns

A reference to the data of the (key,data) pair.

Allows for easy lookup with the subscript ( [] ) operator. Returns data associated with the key specified in subscript. If the key does not exist, a pair with that key is created using default values, which is then returned.

Lookup requires logarithmic time.

Definition at line 445 of file stl\_map.h.

References std::map< \_Key, \_Tp, \_Compare, \_Alloc >::end(), std::map< \_Key, \_Tp, \_Compare, \_Alloc >::insert(), std::map< \_Key, \_Tp, \_Compare, \_Alloc >::key\_comp(), and std::map< \_Key, \_Tp, \_Compare, \_Alloc >::lower\_bound().

**5.586.3.33** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> reverse_iterator std::map< _Key, _Tp, _Compare,  
_Alloc >::rbegin ( ) [inline]`

Returns a read/write reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

Definition at line 345 of file stl\_map.h.

**5.586.3.34** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> const_reverse_iterator std::map< _Key, _Tp,  
_Compare, _Alloc >::rbegin ( ) const [inline]`

Returns a read-only (constant) reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

Definition at line 354 of file stl\_map.h.

**5.586.3.35** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> reverse_iterator std::map< _Key, _Tp, _Compare,  
_Alloc >::rend ( ) [inline]`

Returns a read/write reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

Definition at line 363 of file stl\_map.h.

**5.586.3.36** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> const_reverse_iterator std::map< _Key, _Tp,  
_Compare, _Alloc >::rend ( ) const [inline]`

Returns a read-only (constant) reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

Definition at line 372 of file stl\_map.h.

**5.586.3.37** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> size_type std::map< _Key, _Tp, _Compare, _Alloc  
>::size ( ) const [inline]`

Returns the size of the map.

Definition at line 423 of file stl\_map.h.

**5.586.3.38** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> void std::map< _Key, _Tp, _Compare, _Alloc  
>::swap ( map< _Key, _Tp, _Compare, _Alloc > & __x )  
[inline]`

Swaps data with another map.

### Parameters

*x* A map of the same element and allocator types.

This exchanges the elements between two maps in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(m1,m2)` will feed to this function.

Definition at line 693 of file stl\_map.h.

Referenced by `std::map< _Key, _Tp, _Compare, _Alloc >::operator=()`, and `std::swap()`.

**5.586.3.39** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> iterator std::map< _Key, _Tp, _Compare, _Alloc  
>::upper_bound ( const key_type & __x ) [inline]`

Finds the end of a subsequence matching given key.

#### Parameters

*x* Key of (key, value) pair to be located.

#### Returns

Iterator pointing to the first element greater than key, or [end\(\)](#).

Definition at line 803 of file `stl_map.h`.

**5.586.3.40** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> const_iterator std::map< _Key, _Tp, _Compare,  
_Alloc >::upper_bound ( const key_type & __x ) const  
[inline]`

Finds the end of a subsequence matching given key.

#### Parameters

*x* Key of (key, value) pair to be located.

#### Returns

Read-only (constant) iterator pointing to first iterator greater than key, or [end\(\)](#).

Definition at line 813 of file `stl_map.h`.

**5.586.3.41** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> value_compare std::map< _Key, _Tp, _Compare,  
_Alloc >::value_comp ( ) const [inline]`

Returns a value comparison object, built from the key comparison object out of which the map was constructed.



Definition at line 720 of file stl\_map.h.

The documentation for this class was generated from the following file:

- [stl\\_map.h](#)

## 5.587 std::mask\_array< \_Tp > Class Template Reference

Reference to selected subset of an array.

### Public Types

- typedef \_Tp **value\_type**

### Public Member Functions

- [mask\\_array](#) (const [mask\\_array](#) &)
- template<class \_Dom >  
void **operator%=(const \_Expr< \_Dom, \_Tp > &)** const
- void **operator%=(const [valarray](#)< \_Tp > &)** const
- void **operator&=(const [valarray](#)< \_Tp > &)** const
- template<class \_Dom >  
void **operator&=(const \_Expr< \_Dom, \_Tp > &)** const
- template<class \_Dom >  
void **operator\*=(const \_Expr< \_Dom, \_Tp > &)** const
- void **operator\*=(const [valarray](#)< \_Tp > &)** const
- template<class \_Dom >  
void **operator+=(const \_Expr< \_Dom, \_Tp > &)** const
- void **operator+=(const [valarray](#)< \_Tp > &)** const
- void **operator-=(const [valarray](#)< \_Tp > &)** const
- template<class \_Dom >  
void **operator-=(const \_Expr< \_Dom, \_Tp > &)** const
- template<class \_Dom >  
void **operator/=(const \_Expr< \_Dom, \_Tp > &)** const
- void **operator/=(const [valarray](#)< \_Tp > &)** const
- void **operator<<=(const [valarray](#)< \_Tp > &)** const
- template<class \_Dom >  
void **operator<<=(const \_Expr< \_Dom, \_Tp > &)** const
- template<class \_Dom >  
void **operator=(const \_Expr< \_Dom, \_Tp > &)** const
- [mask\\_array](#) & **operator=(const [mask\\_array](#) &)**
- void **operator=(const [valarray](#)< \_Tp > &)** const

- template<class \_Ex >  
void **operator**= (const \_Expr< \_Ex, \_Tp > &\_\_e) const
- void **operator**= (const \_Tp &) const
- void **operator**>>= (const valarray< \_Tp > &) const
- template<class \_Dom >  
void **operator**>>= (const \_Expr< \_Dom, \_Tp > &) const
- void **operator**^= (const valarray< \_Tp > &) const
- template<class \_Dom >  
void **operator**^= (const \_Expr< \_Dom, \_Tp > &) const
- void **operator**|= (const valarray< \_Tp > &) const
- template<class \_Dom >  
void **operator**|= (const \_Expr< \_Dom, \_Tp > &) const

## Friends

- class valarray< \_Tp >

### 5.587.1 Detailed Description

template<class \_Tp> class std::mask\_array< \_Tp >

Reference to selected subset of an array. A [mask\\_array](#) is a reference to the actual elements of an array specified by a bitmask in the form of an array of bool. The way to get a [mask\\_array](#) is to call operator[] (valarray<bool>) on a valarray. The returned [mask\\_array](#) then permits carrying operations out on the referenced subset of elements in the original valarray.

For example, if a [mask\\_array](#) is obtained using the array (false, true, false, true) as an argument, the mask array has two elements referring to array[1] and array[3] in the underlying array.

## Parameters

*Tp* Element type.

Definition at line 63 of file mask\_array.h.

### 5.587.2 Member Function Documentation

**5.587.2.1** template<class \_Tp> void std::mask\_array< \_Tp >::operator%= (const valarray< \_Tp > & ) const

Modulo slice elements by corresponding elements of v.

**5.587.2.2** `template<class _Tp> void std::mask_array< _Tp >::operator&= (const valarray< _Tp > & ) const`

Logical and slice elements with corresponding elements of *v*.

**5.587.2.3** `template<class _Tp> void std::mask_array< _Tp >::operator*= (const valarray< _Tp > & ) const`

Multiply slice elements by corresponding elements of *v*.

**5.587.2.4** `template<class _Tp> void std::mask_array< _Tp >::operator+= (const valarray< _Tp > & ) const`

Add corresponding elements of *v* to slice elements.

**5.587.2.5** `template<class _Tp> void std::mask_array< _Tp >::operator-= (const valarray< _Tp > & ) const`

Subtract corresponding elements of *v* from slice elements.

**5.587.2.6** `template<class _Tp> void std::mask_array< _Tp >::operator/= (const valarray< _Tp > & ) const`

Divide slice elements by corresponding elements of *v*.

**5.587.2.7** `template<class _Tp> void std::mask_array< _Tp >::operator<<= (const valarray< _Tp > & ) const`

Left shift slice elements by corresponding elements of *v*.

## 5.588 `std::match_results<_Bi_iter, _Allocator>` Class Template Reference

**5.587.2.8** `template<class _Tp> void std::mask_array<_Tp>::operator>>= (const valarray<_Tp> & ) const`

Right shift slice elements by corresponding elements of *v*.

**5.587.2.9** `template<class _Tp> void std::mask_array<_Tp>::operator^= (const valarray<_Tp> & ) const`

Logical xor slice elements with corresponding elements of *v*.

**5.587.2.10** `template<class _Tp> void std::mask_array<_Tp>::operator|= (const valarray<_Tp> & ) const`

Logical or slice elements with corresponding elements of *v*.

The documentation for this class was generated from the following file:

- [mask\\_array.h](#)

## 5.588 `std::match_results<_Bi_iter, _Allocator>` Class Template Reference

The results of a match or search operation.

Inheritance diagram for `std::match_results<_Bi_iter, _Allocator>`:



### Private Types

- `typedef _Tp_alloc_type::const_pointer` **const\_pointer**
- `typedef std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- `typedef _Tp_alloc_type::pointer` **pointer**
- `typedef std::reverse_iterator< iterator >` **reverse\_iterator**

## Private Member Functions

- `_Tp_alloc_type::pointer` **\_M\_allocate** (size\_t \_\_n)
- `pointer` **\_M\_allocate\_and\_copy** (size\_type \_\_n, \_ForwardIterator \_\_first, \_ForwardIterator \_\_last)
- `void` **\_M\_assign\_aux** (\_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
- `void` **\_M\_assign\_aux** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- `void` **\_M\_assign\_dispatch** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- `void` **\_M\_assign\_dispatch** (\_Integer \_\_n, \_Integer \_\_val, \_\_true\_type)
- `size_type` **\_M\_check\_len** (size\_type \_\_n, const char \*\_\_s) const
- `void` **\_M\_deallocate** (typename \_Tp\_alloc\_type::pointer \_\_p, size\_t \_\_n)
- `void` **\_M\_default\_append** (size\_type \_\_n)
- `void` **\_M\_default\_initialize** (size\_type \_\_n)
- `void` **\_M\_erase\_at\_end** (pointer \_\_pos)
- `void` **\_M\_fill\_assign** (size\_type \_\_n, const [value\\_type](#) &\_\_val)
- `void` **\_M\_fill\_initialize** (size\_type \_\_n, const [value\\_type](#) &\_\_value)
- `void` **\_M\_fill\_insert** (iterator \_\_pos, size\_type \_\_n, const [value\\_type](#) &\_\_x)
- `const` `_Tp_alloc_type` & **\_M\_get\_Tp\_allocator** () const
- `_Tp_alloc_type` & **\_M\_get\_Tp\_allocator** ()
- `void` **\_M\_initialize\_dispatch** (\_Integer \_\_n, \_Integer \_\_value, \_\_true\_type)
- `void` **\_M\_initialize\_dispatch** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- `void` **\_M\_insert\_aux** (iterator \_\_position, \_Args &&...\_\_args)
- `void` **\_M\_insert\_dispatch** (iterator \_\_pos, \_Integer \_\_n, \_Integer \_\_val, \_\_true\_type)
- `void` **\_M\_insert\_dispatch** (iterator \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- `void` **\_M\_range\_check** (size\_type \_\_n) const
- `void` **\_M\_range\_initialize** (\_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
- `void` **\_M\_range\_initialize** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- `void` **\_M\_range\_insert** (iterator \_\_pos, \_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- `void` **\_M\_range\_insert** (iterator \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
- `void` **assign** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- `void` **assign** (size\_type \_\_n, const [value\\_type](#) &\_\_val)
- `void` **assign** ([initializer\\_list](#)< [value\\_type](#) > \_\_l)
- [reference](#) at (size\_type \_\_n)
- [const\\_reference](#) at (size\_type \_\_n) const

- `reference back()`
- `const_reference back()` const
- iterator `begin()`
- `size_type capacity()` const
- void `clear()`
- `const_reverse_iterator crbegin()` const
- `const_reverse_iterator crend()` const
- `std::sub_match<_Bi_iter> * data()`
- const `std::sub_match<_Bi_iter> * data()` const
- iterator `emplace(iterator __position, _Args &&... __args)`
- void `emplace_back(_Args &&... __args)`
- iterator `end()`
- iterator `erase(iterator __position)`
- iterator `erase(iterator __first, iterator __last)`
- `reference front()`
- `const_reference front()` const
- void `insert(iterator __position, _InputIterator __first, _InputIterator __last)`
- iterator `insert(iterator __position, value_type && __x)`
- void `insert(iterator __position, size_type __n, const value_type & __x)`
- iterator `insert(iterator __position, const value_type & __x)`
- void `insert(iterator __position, initializer_list<value_type> __l)`
- `reference operator[] (size_type __n)`
- `const_reference operator[] (size_type __n)` const
- void `pop_back()`
- void `push_back(const value_type & __x)`
- void `push_back(value_type && __x)`
- `const_reverse_iterator rbegin()` const
- `reverse_iterator rbegin()`
- `reverse_iterator rend()`
- `const_reverse_iterator rend()` const
- void `reserve(size_type __n)`
- void `resize(size_type __new_size, const value_type & __x)`
- void `resize(size_type __new_size)`
- void `shrink_to_fit()`
- void `swap(vector & __x)`

#### Private Attributes

- `_Vector_impl _M_impl`

#### Friends

- class `__regex::SpecializedResults<_Bi_iter, _Allocator>`

## 10.? Public Types

- typedef `sub_match<_Bi_iter>` `value_type`
- typedef const `value_type` & `const_reference`
- typedef `const_reference` `reference`
- typedef `_Base_type::const_iterator` `const_iterator`
- typedef const\_iterator `iterator`
- typedef `std::iterator_traits<_Bi_iter>::difference_type` `difference_type`
- typedef `_Allocator::size_type` `size_type`
- typedef `_Allocator` `allocator_type`
- typedef `std::iterator_traits<_Bi_iter>::value_type` `char_type`
- typedef `std::basic_string<char_type>` `string_type`

## 10.1 Construction, Copying, and Destruction

- `match_results` (const `_Allocator` &\_\_a=`_Allocator`())
- `match_results` (const `match_results` &\_\_rhs)
- `match_results` & `operator=` (const `match_results` \_\_rhs)
- `~match_results` ()

## 10.2 Size

- `size_type` `size` () const
- `size_type` `max_size` () const
- bool `empty` () const

## 10.3 Element Access

- `difference_type` `length` (`size_type` \_\_sub=0) const
- `difference_type` `position` (`size_type` \_\_sub=0) const
- `string_type` `str` (`size_type` \_\_sub=0) const
- `const_reference` `operator[]` (`size_type` \_\_sub) const
- `const_reference` `prefix` () const
- `const_reference` `suffix` () const
- const\_iterator `begin` () const
- const\_iterator `cbegin` () const
- const\_iterator `end` () const
- const\_iterator `cend` () const

## 10.4 Formatting

These functions perform formatted substitution of the matched character sequences into their target. The format specifiers and escape sequences accepted by these functions are determined by their `flags` parameter as documented above.

- `template<typename _Out_iter>`  
`_Out_iter format (_Out_iter __out, const string_type &__fmt, regex_constants::match_flag_type __flags=regex_constants::format_default) const`
- `string_type format (const string_type &__fmt, regex_constants::match_flag_type __flags=regex_constants::format_default) const`

## 10.5 Allocator

- `allocator_type get_allocator () const`

## 10.6 Swap

- `void swap (match_results &__that)`

### 5.588.1 Detailed Description

`template<typename _Bi_iter, typename _Allocator = allocator<sub_match<_Bi_iter>>> class std::match_results<_Bi_iter, _Allocator>`

The results of a match or search operation. A collection of character sequences representing the result of a regular expression match. Storage for the collection is allocated and freed as necessary by the member functions of class template `match_results`.

This class satisfies the Sequence requirements, with the exception that only the operations defined for a const-qualified Sequence are supported.

The `sub_match` object stored at index 0 represents sub-expression 0, i.e. the whole match. In this case the `sub_match` member `matched` is always true. The `sub_match` object stored at index `n` denotes what matched the marked sub-expression `n` within the matched expression. If the sub-expression `n` participated in a regular expression match then the `sub_match` member `matched` evaluates to true, and members `first` and `second` denote the range of characters [`first`, `second`) which formed that match. Otherwise `matched` is false, and members `first` and `second` point to the end of the sequence that was searched.

Definition at line 1446 of file `regex.h`.



## 5.588.2 Constructor & Destructor Documentation

**5.588.2.1** `template<typename _Bi_iter, typename _Allocator =  
allocator<sub_match<_Bi_iter>>> std::match_results<  
_Bi_iter, _Allocator>::match_results ( const _Allocator & __a =  
_Allocator() ) [inline, explicit]`

Constructs a default match\_results container.

### Postcondition

`size()` returns 0 and `str()` returns an empty string.

Definition at line 1495 of file regex.h.

Referenced by `std::match_results<_FwdIterT, _Alloc>::operator=()`.

**5.588.2.2** `template<typename _Bi_iter, typename _Allocator =  
allocator<sub_match<_Bi_iter>>> std::match_results<_Bi_iter,  
_Allocator>::match_results ( const match_results<_Bi_iter,  
_Allocator> & __rhs ) [inline]`

Copy constructs a match\_results.

Definition at line 1502 of file regex.h.

**5.588.2.3** `template<typename _Bi_iter, typename _Allocator =  
allocator<sub_match<_Bi_iter>>> std::match_results<_Bi_iter,  
_Allocator>::~~match_results ( ) [inline]`

Destroys a match\_results object.

Definition at line 1519 of file regex.h.

## 5.588.3 Member Function Documentation

**5.588.3.1** `template<typename _Bi_iter, typename _Allocator =  
allocator<sub_match<_Bi_iter>>> const_iterator  
std::match_results<_Bi_iter, _Allocator>::begin ( ) const  
[inline]`

## 5.588 `std::match_results<_Bi_iter, _Allocator>` Class Template Reference 2919

---

Gets an iterator to the start of the `sub_match` collection.

Reimplemented from `std::vector< std::sub_match< _Bi_iter >, _Allocator >`.

Definition at line 1658 of file `regex.h`.

```
5.588.3.2 template<typename _Bi_iter, typename _Allocator =
 allocator<sub_match<_Bi_iter> >> const_iterator
 std::match_results< _Bi_iter, _Allocator >::cbegin () const
 [inline]
```

Gets an iterator to the start of the `sub_match` collection.

Reimplemented from `std::vector< std::sub_match< _Bi_iter >, _Allocator >`.

Definition at line 1665 of file `regex.h`.

```
5.588.3.3 template<typename _Bi_iter, typename _Allocator =
 allocator<sub_match<_Bi_iter> >> const_iterator
 std::match_results< _Bi_iter, _Allocator >::cend () const
 [inline]
```

Gets an iterator to one-past-the-end of the collection.

Reimplemented from `std::vector< std::sub_match< _Bi_iter >, _Allocator >`.

Definition at line 1683 of file `regex.h`.

```
5.588.3.4 template<typename _Bi_iter, typename _Allocator =
 allocator<sub_match<_Bi_iter> >> bool std::match_results<
 _Bi_iter, _Allocator >::empty () const [inline]
```

Indicates if the `match_results` contains no results.

### Return values

*true* The `match_results` object is empty.

*false* The `match_results` object is not empty.

Reimplemented from `std::vector< std::sub_match< _Bi_iter >, _Allocator >`.

Definition at line 1555 of file `regex.h`.

## 5.588 `std::match_results<_Bi_iter, _Allocator>` Class Template Reference 2920

Referenced by `std::match_results<_FwdIterT, _Alloc>::cend()`, `std::match_results<_FwdIterT, _Alloc>::end()`, `std::match_results<_FwdIterT, _Alloc>::prefix()`, and `std::match_results<_FwdIterT, _Alloc>::suffix()`.

**5.588.3.5** `template<typename _Bi_iter, typename _Allocator = allocator<sub_match<_Bi_iter>>> const_iterator std::match_results<_Bi_iter, _Allocator>::end ( ) const [inline]`

Gets an iterator to one-past-the-end of the collection.

Reimplemented from `std::vector< std::sub_match<_Bi_iter>, _Allocator>`.

Definition at line 1672 of file `regex.h`.

**5.588.3.6** `template<typename _Bi_iter, typename _Allocator = allocator<sub_match<_Bi_iter>>> string_type std::match_results<_Bi_iter, _Allocator>::format ( const string_type & __fmt, regex_constants::match_flag_type __flags = regex_constants::format_default ) const`

### Todo

Implement this function.

**5.588.3.7** `template<typename _Bi_iter, typename _Allocator = allocator<sub_match<_Bi_iter>>> template<typename _Out_iter > _Out_iter std::match_results<_Bi_iter, _Allocator>::format ( _Out_iter __out, const string_type & __fmt, regex_constants::match_flag_type __flags = regex_constants::format_default ) const [inline]`

### Todo

Implement this function.

Definition at line 1707 of file `regex.h`.

## 5.588 std::match\_results<\_Bi\_iter, \_Allocator> Class Template Reference 2921

---

**5.588.3.8** `template<typename _Bi_iter, typename _Allocator =  
allocator<sub_match<_Bi_iter>>> allocator_type  
std::match_results<_Bi_iter, _Allocator>::get_allocator ( ) const  
[inline]`

Gets a copy of the allocator.

Reimplemented from `std::_Vector_base< std::sub_match<_Bi_iter>, _Allocator>`.

Definition at line 1731 of file `regex.h`.

**5.588.3.9** `template<typename _Bi_iter, typename _Allocator =  
allocator<sub_match<_Bi_iter>>> difference_type  
std::match_results<_Bi_iter, _Allocator>::length ( size_type __sub  
= 0 ) const [inline]`

Gets the length of the indicated submatch.

### Parameters

*sub* indicates the submatch.

This function returns the length of the indicated submatch, or the length of the entire match if *sub* is zero (the default).

Definition at line 1573 of file `regex.h`.

**5.588.3.10** `template<typename _Bi_iter, typename _Allocator =  
allocator<sub_match<_Bi_iter>>> size_type std::match_results<  
_Bi_iter, _Allocator>::max_size ( ) const [inline]`

Gets the number of matches and submatches.

The number of matches for a given regular expression will be either 0 if there was no match or `mark_count() + 1` if a match was successful. Some matches may be empty.

### Returns

the number of matches found.

Reimplemented from `std::vector< std::sub_match<_Bi_iter>, _Allocator>`.

Definition at line 1546 of file `regex.h`.

## 5.588 `std::match_results<_Bi_iter, _Allocator>` Class Template Reference 2922

---

**5.588.3.11** `template<typename _Bi_iter, typename _Allocator =  
allocator<sub_match<_Bi_iter>>> match_results&  
std::match_results<_Bi_iter, _Allocator>::operator= ( const  
match_results<_Bi_iter, _Allocator> __rhs ) [inline]`

Assigns rhs to \*this.

Definition at line 1510 of file regex.h.

**5.588.3.12** `template<typename _Bi_iter, typename _Allocator =  
allocator<sub_match<_Bi_iter>>> const_reference  
std::match_results<_Bi_iter, _Allocator>::operator[] ( size_type  
__sub ) const [inline]`

Gets a sub\_match reference for the match or submatch.

### Parameters

*sub* indicates the submatch.

This function gets a reference to the indicated submatch, or the entire match if *sub* is zero.

If *sub* >= `size()` then this function returns a sub\_match with a special value indicating no submatch.

Definition at line 1617 of file regex.h.

**5.588.3.13** `template<typename _Bi_iter, typename _Allocator =  
allocator<sub_match<_Bi_iter>>> difference_type  
std::match_results<_Bi_iter, _Allocator>::position ( size_type  
__sub = 0 ) const [inline]`

Gets the offset of the beginning of the indicated submatch.

### Parameters

*sub* indicates the submatch.

This function returns the offset from the beginning of the target sequence to the beginning of the submatch, unless the value of *sub* is zero (the default), in which case this

## 5.588 `std::match_results<_Bi_iter, _Allocator>` Class Template Reference 2923

function returns the offset from the beginning of the target sequence to the beginning of the match.

Returns -1 if `sub` is out of range.

Definition at line 1589 of file `regex.h`.

**5.588.3.14** `template<typename _Bi_iter, typename _Allocator = allocator<sub_match<_Bi_iter>>> const_reference std::match_results<_Bi_iter, _Allocator>::prefix ( ) const [inline]`

Gets a `sub_match` representing the match prefix.

This function gets a reference to a `sub_match` object representing the part of the target range between the start of the target range and the start of the match.

Definition at line 1632 of file `regex.h`.

Referenced by `std::match_results<_FwdIterT, _Alloc>::position()`.

**5.588.3.15** `template<typename _Bi_iter, typename _Allocator = allocator<sub_match<_Bi_iter>>> size_type std::match_results<_Bi_iter, _Allocator>::size ( ) const [inline]`

Gets the number of matches and submatches.

The number of matches for a given regular expression will be either 0 if there was no match or `mark_count() + 1` if a match was successful. Some matches may be empty.

### Returns

the number of matches found.

Reimplemented from `std::vector< std::sub_match<_Bi_iter>, _Allocator>`.

Definition at line 1539 of file `regex.h`.

Referenced by `std::match_results<_FwdIterT, _Alloc>::operator[]()`, and `std::match_results<_FwdIterT, _Alloc>::position()`.

## 5.588 `std::match_results<_Bi_iter, _Allocator>` Class Template Reference 2924

**5.588.3.16** `template<typename _Bi_iter, typename _Allocator  
= allocator<sub_match<_Bi_iter>>> string_type  
std::match_results<_Bi_iter, _Allocator>::str ( size_type __sub =  
0 ) const [inline]`

Gets the match or submatch converted to a string type.

### Parameters

*sub* indicates the submatch.

This function gets the submatch (or match, if *sub* is zero) extracted from the target range and converted to the associated string type.

Definition at line 1603 of file `regex.h`.

**5.588.3.17** `template<typename _Bi_iter, typename _Allocator =  
allocator<sub_match<_Bi_iter>>> const_reference  
std::match_results<_Bi_iter, _Allocator>::suffix ( ) const  
[inline]`

Gets a `sub_match` representing the match suffix.

This function gets a reference to a `sub_match` object representing the part of the target range between the end of the match and the end of the target range.

Definition at line 1647 of file `regex.h`.

**5.588.3.18** `template<typename _Bi_iter, typename _Allocator =  
allocator<sub_match<_Bi_iter>>> void std::match_results<  
_Bi_iter, _Allocator>::swap ( match_results<_Bi_iter, _Allocator  
> & __that ) [inline]`

Swaps the contents of two [match\\_results](#).

Definition at line 1745 of file `regex.h`.

Referenced by `std::swap()`.

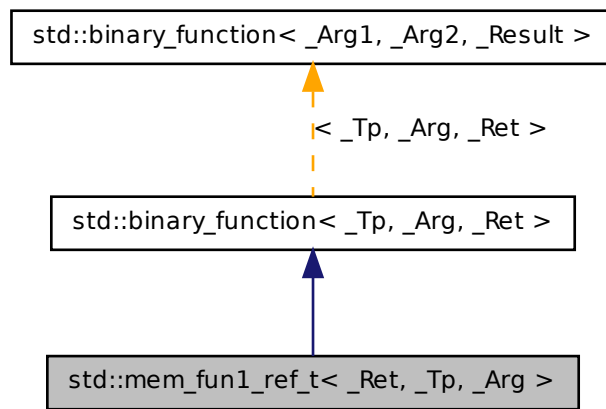
The documentation for this class was generated from the following file:

- [regex.h](#)

## 5.589 std::mem\_fun1\_ref\_t< \_Ret, \_Tp, \_Arg > Class Template Reference

One of the [adaptors for member /// pointers](#).

Inheritance diagram for std::mem\_fun1\_ref\_t< \_Ret, \_Tp, \_Arg >:



### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `_Ret` [result\\_type](#)
- typedef `_Arg` [second\\_argument\\_type](#)

### Public Member Functions

- `mem_fun1_ref_t` (`_Ret` (`_Tp::*__pf`) (`_Arg`))
- `_Ret operator()` (`_Tp &__r, _Arg __x`) const



### 5.589.1 Detailed Description

`template<typename _Ret, typename _Tp, typename _Arg> class std::mem_fun1_ref_t<_Ret, _Tp, _Arg>`

One of the [adaptors for member /// pointers](#).

Definition at line 650 of file `stl_function.h`.

### 5.589.2 Member Typedef Documentation

**5.589.2.1** `typedef _Tp std::binary_function<_Tp, _Arg, _Ret>::first_argument_type [inherited]`

`first_argument_type` is the type of the first argument

Definition at line 118 of file `stl_function.h`.

**5.589.2.2** `typedef _Ret std::binary_function<_Tp, _Arg, _Ret>::result_type [inherited]`

`result_type` is the return type

Definition at line 124 of file `stl_function.h`.

**5.589.2.3** `typedef _Arg std::binary_function<_Tp, _Arg, _Ret>::second_argument_type [inherited]`

`second_argument_type` is the type of the second argument

Definition at line 121 of file `stl_function.h`.

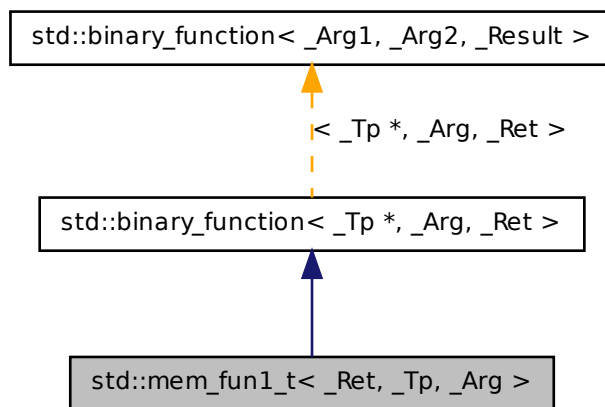
The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 5.590 `std::mem_fun1_t<_Ret, _Tp, _Arg>` Class Template Reference

One of the [adaptors for member /// pointers](#).

Inheritance diagram for std::mem\_fun1\_t< \_Ret, \_Tp, \_Arg >:



### Public Types

- typedef \_Tp \* [first\\_argument\\_type](#)
- typedef \_Ret [result\\_type](#)
- typedef \_Arg [second\\_argument\\_type](#)

### Public Member Functions

- **mem\_fun1\_t** (\_Ret(\_Tp::\*\_\_pf)(\_Arg))
- **\_Ret operator()** (\_Tp \*\_\_p, \_Arg \_\_x) const

#### 5.590.1 Detailed Description

**template<typename \_Ret, typename \_Tp, typename \_Arg> class std::mem\_fun1\_t< \_Ret, \_Tp, \_Arg >**

One of the [adaptors for member /// pointers](#).

Definition at line 614 of file stl\_function.h.

## 5.590.2 Member Typedef Documentation

**5.590.2.1** `typedef _Tp * std::binary_function< _Tp *, _Arg , _Ret  
>::first_argument_type [inherited]`

`first_argument_type` is the type of the first argument

Definition at line 118 of file `stl_function.h`.

**5.590.2.2** `typedef _Ret std::binary_function< _Tp *, _Arg , _Ret  
>::result_type [inherited]`

`result_type` is the return type

Definition at line 124 of file `stl_function.h`.

**5.590.2.3** `typedef _Arg std::binary_function< _Tp *, _Arg , _Ret  
>::second_argument_type [inherited]`

`second_argument_type` is the type of the second argument

Definition at line 121 of file `stl_function.h`.

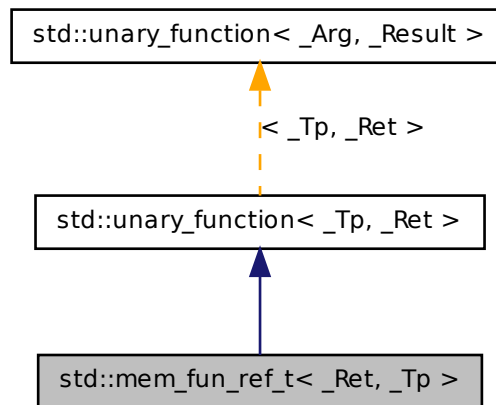
The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 5.591 `std::mem_fun_ref_t<_Ret, _Tp>` Class Template Reference

One of the [adaptors for member /// pointers](#).

Inheritance diagram for `std::mem_fun_ref_t<_Ret, _Tp>`:



### Public Types

- `typedef _Tp` [argument\\_type](#)
- `typedef _Ret` [result\\_type](#)

### Public Member Functions

- `mem_fun_ref_t` (`_Ret`(`_Tp::*_pf`)())
- `_Ret operator()` (`_Tp &__r`) `const`

#### 5.591.1 Detailed Description

`template<typename _Ret, typename _Tp> class std::mem_fun_ref_t<_Ret, _Tp>`

One of the [adaptors for member /// pointers](#).

Definition at line 578 of file `stl_function.h`.

### 5.591.2 Member Typedef Documentation

#### 5.591.2.1 `typedef _Tp std::unary_function<_Tp, _Ret>::argument_type` [`inherited`]

`argument_type` is the type of the argument

Definition at line 105 of file `stl_function.h`.

#### 5.591.2.2 `typedef _Ret std::unary_function<_Tp, _Ret>::result_type` [`inherited`]

`result_type` is the return type

Definition at line 108 of file `stl_function.h`.

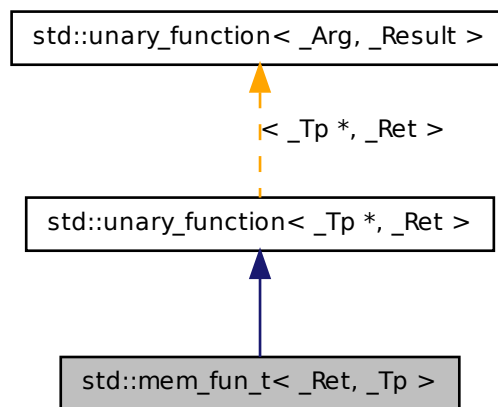
The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

### 5.592 `std::mem_fun_t<_Ret, _Tp>` Class Template Reference

One of the [adaptors for member /// pointers](#).

Inheritance diagram for std::mem\_fun\_t< \_Ret, \_Tp >:



### Public Types

- typedef `_Tp *` [argument\\_type](#)
- typedef `_Ret` [result\\_type](#)

### Public Member Functions

- `mem_fun_t` (`_Ret`(`_Tp::*_pf`)())
- `_Ret operator()` (`_Tp *_p`) const

#### 5.592.1 Detailed Description

**template**<typename `_Ret`, typename `_Tp`> class `std::mem_fun_t`< `_Ret`, `_Tp` >

One of the [adaptors for member /// pointers](#).

Definition at line 542 of file `stl_function.h`.

### 5.592.2 Member Typedef Documentation

#### 5.592.2.1 `typedef _Tp * std::unary_function<_Tp *, _Ret>::argument_type` [`inherited`]

`argument_type` is the type of the argument

Definition at line 105 of file `stl_function.h`.

#### 5.592.2.2 `typedef _Ret std::unary_function<_Tp *, _Ret>::result_type` [`inherited`]

`result_type` is the return type

Definition at line 108 of file `stl_function.h`.

The documentation for this class was generated from the following file:

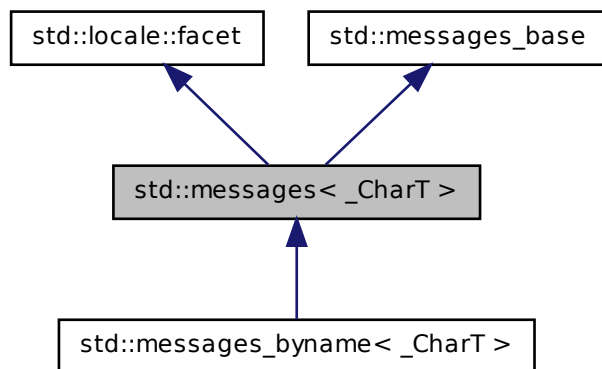
- [stl\\_function.h](#)

## 5.593 `std::messages<_CharT>` Class Template Reference

Primary class template messages.

This facet encapsulates the code to retrieve messages from message catalogs. The only thing defined by the standard for this facet is the interface. All underlying functionality is implementation-defined.

Inheritance diagram for `std::messages<_CharT>`:



### Public Types

- typedef int **catalog**
- typedef `_CharT` **char\_type**
- typedef `basic_string<_CharT>` **string\_type**

### Public Member Functions

- **messages** (size\_t \_\_refs=0)
- **messages** (\_\_c\_locale \_\_cloc, const char \*\_\_s, size\_t \_\_refs=0)
- void **close** (catalog \_\_c) const
- **string\_type** **get** (catalog \_\_c, int \_\_set, int \_\_msgid, const **string\_type** &\_\_s) const
- catalog **open** (const **basic\_string**< char > &\_\_s, const **locale** &\_\_loc) const
- catalog **open** (const **basic\_string**< char > &, const **locale** &, const char \*) const

### Static Public Attributes

- static **locale::id** **id**



### Protected Member Functions

- virtual `~messages()`
- `string_type _M_convert_from_char` (char \*) const
- char \* `_M_convert_to_char` (const `string_type` &\_\_msg) const
- virtual void `do_close` (catalog) const
- template<>  
  `string do_get` (catalog, int, int, const `string` &) const
- template<>  
  `wstring do_get` (catalog, int, int, const `wstring` &) const
- virtual `string_type do_get` (catalog, int, int, const `string_type` &\_\_dfault) const
- virtual catalog `do_open` (const `basic_string`< char > &, const `locale` &) const

### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale` (\_\_c\_locale &\_\_cloc) throw ()
- static void `_S_create_c_locale` (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void `_S_destroy_c_locale` (\_\_c\_locale &\_\_cloc)
- static `__c_locale _S_get_c_locale` ()
- static const char \* `_S_get_c_name` () throw ()
- static `__c_locale _S_lc_ctype_c_locale` (\_\_c\_locale \_\_cloc, const char \*\_\_s)

### Protected Attributes

- `__c_locale _M_c_locale_messages`
- const char \* `_M_name_messages`

### Friends

- class `locale::Impl`

#### 5.593.1 Detailed Description

**template<typename \_CharT> class std::messages<\_CharT>**

Primary class template messages.

This facet encapsulates the code to retrieve messages from message catalogs. The only thing defined by the standard for this facet is the interface. All underlying functionality is implementation-defined. This library currently implements 3 versions of the message facet. The first version (gnu) is a wrapper around gettext, provided by libintl. The second version (ieee) is a wrapper around catgets. The final version (default) does

no actual translation. These implementations are only provided for `char` and `wchar_t` instantiations.

The messages template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the messages facet.

Definition at line 1695 of file `locale_facets_nonio.h`.

### 5.593.2 Member Typedef Documentation

#### 5.593.2.1 `template<typename _CharT> typedef _CharT std::messages<_CharT>::char_type`

Public typedefs.

Reimplemented in [std::messages\\_byname<\\_CharT>](#).

Definition at line 1701 of file `locale_facets_nonio.h`.

#### 5.593.2.2 `template<typename _CharT> typedef basic_string<_CharT> std::messages<_CharT>::string_type`

Public typedefs.

Reimplemented in [std::messages\\_byname<\\_CharT>](#).

Definition at line 1702 of file `locale_facets_nonio.h`.

### 5.593.3 Constructor & Destructor Documentation

#### 5.593.3.1 `template<typename _CharT> std::messages<_CharT>::messages ( size_t __refs = 0 ) [explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

#### Parameters

*refs* Passed to the base facet class.

Definition at line 45 of file `messages_members.h`.

**5.593.3.2** `template<typename _CharT > std::messages< _CharT >::messages  
( __c_locale __cloc, const char * __s, size_t __refs = 0 )  
[explicit]`

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

#### Parameters

*cloc* The C locale.

*s* The name of a locale.

*refs* Refcount to pass to the base class.

Definition at line 51 of file messages\_members.h.

**5.593.3.3** `template<typename _CharT > std::messages< _CharT  
>::~messages ( ) [protected, virtual]`

Destructor.

Definition at line 80 of file messages\_members.h.

#### 5.593.4 Member Data Documentation

**5.593.4.1** `template<typename _CharT > locale::id std::messages< _CharT  
>::id [static]`

Numpunct facet id.

Definition at line 1713 of file locale\_facets\_nonio.h.

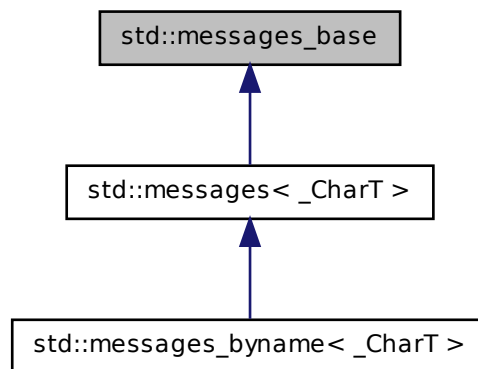
The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [messages\\_members.h](#)

## 5.594 std::messages\_base Struct Reference

Messages facet base class providing catalog typedef.

Inheritance diagram for `std::messages_base`:



### Public Types

- typedef int **catalog**

#### 5.594.1 Detailed Description

Messages facet base class providing catalog typedef.

Definition at line 1668 of file `locale_facets_nonio.h`.

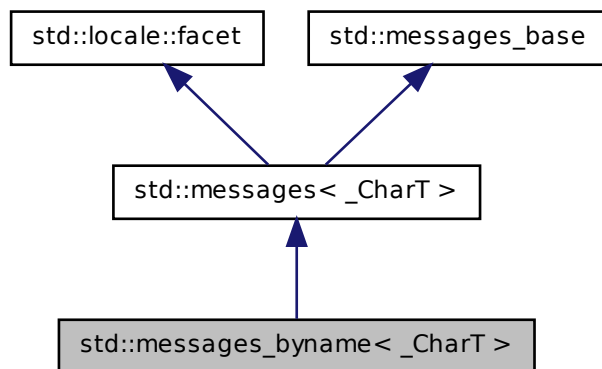
The documentation for this struct was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

### 5.595 `std::messages_byname<_CharT>` Class Template Reference

class [messages\\_byname](#) [22.2.7.2].

Inheritance diagram for `std::messages_byname<_CharT>`:



### Public Types

- typedef int **catalog**
- typedef `_CharT` **char\_type**
- typedef `basic_string<_CharT>` **string\_type**

### Public Member Functions

- **messages\_byname** (const char \*\_\_s, size\_t \_\_refs=0)
- void **close** (catalog \_\_c) const
- **string\_type** **get** (catalog \_\_c, int \_\_set, int \_\_msgid, const **string\_type** &\_\_s) const
- catalog **open** (const `basic_string<char>` &, const **locale** &, const char \*) const
- catalog **open** (const `basic_string<char>` &\_\_s, const **locale** &\_\_loc) const

### Static Public Attributes

- static **locale::id** **id**

### Protected Member Functions

- [string\\_type](#) **\_M\_convert\_from\_char** (char \*) const
- char \* **\_M\_convert\_to\_char** (const [string\\_type](#) &\_\_msg) const
- virtual void **do\_close** (catalog) const
- template<>  
[wstring](#) **do\_get** (catalog, int, int, const [wstring](#) &) const
- template<>  
[string](#) **do\_get** (catalog, int, int, const [string](#) &) const
- virtual [string\\_type](#) **do\_get** (catalog, int, int, const [string\\_type](#) &\_\_default) const
- virtual catalog **do\_open** (const [basic\\_string](#)< char > &, const [locale](#) &) const

### Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

### Protected Attributes

- \_\_c\_locale **\_M\_c\_locale\_messages**
- const char \* **\_M\_name\_messages**

### Friends

- class **locale::Impl**

#### 5.595.1 Detailed Description

template<typename \_CharT> class std::messages\_byname<\_CharT>

class [messages\\_byname](#) [22.2.7.2].

Definition at line 1912 of file locale\_facets\_nonio.h.

### 5.595.2 Member Typedef Documentation

#### 5.595.2.1 `template<typename _CharT> typedef _CharT std::messages_byname<_CharT>::char_type`

Public typedefs.

Reimplemented from [std::messages<\\_CharT>](#).

Definition at line 1915 of file `locale_facets_nonio.h`.

#### 5.595.2.2 `template<typename _CharT> typedef basic_string<_CharT> std::messages_byname<_CharT>::string_type`

Public typedefs.

Reimplemented from [std::messages<\\_CharT>](#).

Definition at line 1916 of file `locale_facets_nonio.h`.

### 5.595.3 Member Data Documentation

#### 5.595.3.1 `template<typename _CharT> locale::id std::messages<_CharT> >::id [static, inherited]`

Numpunct facet id.

Definition at line 1713 of file `locale_facets_nonio.h`.

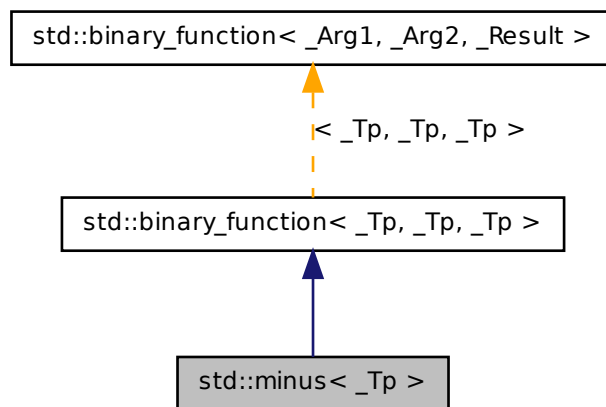
The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [messages\\_members.h](#)

## 5.596 `std::minus<_Tp>` Struct Template Reference

One of the [math functors](#).

Inheritance diagram for std::minus< \_Tp >:



### Public Types

- typedef `_Tp` `first_argument_type`
- typedef `_Tp` `result_type`
- typedef `_Tp` `second_argument_type`

### Public Member Functions

- `_Tp operator() (const _Tp &__x, const _Tp &__y) const`

#### 5.596.1 Detailed Description

**template<typename \_Tp> struct std::minus< \_Tp >**

One of the [math functors](#).

Definition at line 150 of file `stl_function.h`.



### 5.596.2 Member Typedef Documentation

**5.596.2.1** `typedef _Tp std::binary_function< _Tp , _Tp , _Tp  
>::first_argument_type [inherited]`

`first_argument_type` is the type of the first argument

Definition at line 118 of file `stl_function.h`.

**5.596.2.2** `typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::result_type  
[inherited]`

`result_type` is the return type

Definition at line 124 of file `stl_function.h`.

**5.596.2.3** `typedef _Tp std::binary_function< _Tp , _Tp , _Tp  
>::second_argument_type [inherited]`

`second_argument_type` is the type of the second argument

Definition at line 121 of file `stl_function.h`.

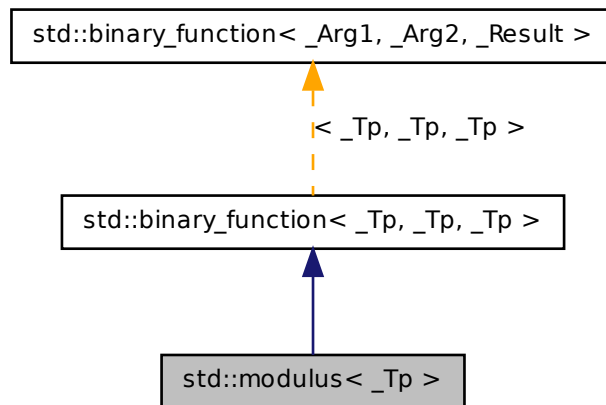
The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.597 `std::modulus<_Tp>` Struct Template Reference

One of the [math functors](#).

Inheritance diagram for std::modulus< \_Tp >:



### Public Types

- typedef `_Tp` `first_argument_type`
- typedef `_Tp` `result_type`
- typedef `_Tp` `second_argument_type`

### Public Member Functions

- `_Tp operator() (const _Tp &__x, const _Tp &__y) const`

#### 5.597.1 Detailed Description

**template<typename \_Tp> struct std::modulus< \_Tp >**

One of the [math functors](#).

Definition at line 177 of file `stl_function.h`.

### 5.597.2 Member Typedef Documentation

**5.597.2.1** `typedef _Tp std::binary_function< _Tp , _Tp , _Tp  
>::first_argument_type [inherited]`

`first_argument_type` is the type of the first argument

Definition at line 118 of file `stl_function.h`.

**5.597.2.2** `typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::result_type  
[inherited]`

`result_type` is the return type

Definition at line 124 of file `stl_function.h`.

**5.597.2.3** `typedef _Tp std::binary_function< _Tp , _Tp , _Tp  
>::second_argument_type [inherited]`

`second_argument_type` is the type of the second argument

Definition at line 121 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

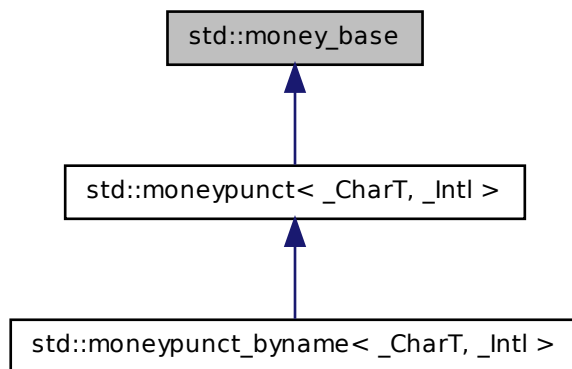
- [stl\\_function.h](#)

## 5.598 `std::money_base` Class Reference

Money format ordering data.

This class contains an ordered array of 4 fields to represent the pattern for formatting a money amount. Each field may contain one entry from the `part` enum. `symbol`, `sign`, and `value` must be present and the remaining field must contain either none or space.

Inheritance diagram for std::money\_base:



### Public Types

- enum { **\_S\_minus**, **\_S\_zero**, **\_S\_end** }
- enum **part** {  
    **none**, **space**, **symbol**, **sign**,  
    **value** }

### Static Public Member Functions

- static pattern **\_S\_construct\_pattern** (char \_\_precedes, char \_\_space, char \_\_posn) throw ()

### Static Public Attributes

- static const char \* **\_S\_atoms**
- static const pattern **\_S\_default\_pattern**

#### 5.598.1 Detailed Description

Money format ordering data.

This class contains an ordered array of 4 fields to represent the pattern for formatting a money amount. Each field may contain one entry from the part enum. symbol, sign, and value must be present and the remaining field must contain either none or space.

**See also**

[moneypunct::pos\\_format\(\)](#) and [moneypunct::neg\\_format\(\)](#) for details of how these fields are interpreted.

Definition at line 840 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following file:

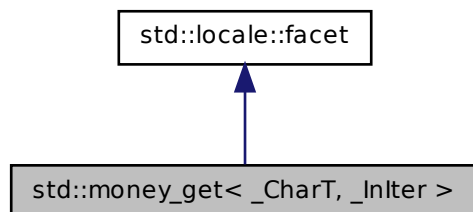
- [locale\\_facets\\_nonio.h](#)

## 5.599 `std::money_get< _CharT, _InIter >` Class Template Reference

Primary class template [money\\_get](#).

This facet encapsulates the code to parse and return a monetary amount from a string.

Inheritance diagram for `std::money_get< _CharT, _InIter >`:

**Public Types**

- typedef `_CharT` [char\\_type](#)
- typedef `_InIter` [iter\\_type](#)
- typedef [basic\\_string< \\_CharT >](#) [string\\_type](#)

### Public Member Functions

- `money_get` (`size_t __refs=0`)
- `iter_type get` (`iter_type __s, iter_type __end, bool __intl, ios_base &__io, ios_base::iostate &__err, string_type &__digits`) const
- `iter_type get` (`iter_type __s, iter_type __end, bool __intl, ios_base &__io, ios_base::iostate &__err, long double &__units`) const

### Static Public Attributes

- static `locale::id id`

### Protected Member Functions

- virtual `~money_get` ()
- `template<bool _Intl>`  
`iter_type _M_extract` (`iter_type __s, iter_type __end, ios_base &__io, ios_base::iostate &__err, string &__digits`) const
- virtual `iter_type do_get` (`iter_type __s, iter_type __end, bool __intl, ios_base &__io, ios_base::iostate &__err, string_type &__digits`) const
- virtual `iter_type do_get` (`iter_type __s, iter_type __end, bool __intl, ios_base &__io, ios_base::iostate &__err, long double &__units`) const

### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale` (`__c_locale &__cloc`) throw ()
- static void `_S_create_c_locale` (`__c_locale &__cloc, const char *__s, __c_locale __old=0`)
- static void `_S_destroy_c_locale` (`__c_locale &__cloc`)
- static `__c_locale _S_get_c_locale` ()
- static const char \* `_S_get_c_name` () throw ()
- static `__c_locale _S_lc_ctype_c_locale` (`__c_locale __cloc, const char *__s`)

### Friends

- class `locale::Impl`

#### 5.599.1 Detailed Description

`template<typename _CharT, typename _InIter> class std::money_get<_CharT, _InIter>`

Primary class template `money_get`.

This facet encapsulates the code to parse and return a monetary amount from a string. The [money\\_get](#) template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the [money\\_get](#) facet.

Definition at line 1370 of file locale\_facets\_nonio.h.

### **5.599.2 Member Typedef Documentation**

#### **5.599.2.1 template<typename \_CharT, typename \_InIter> typedef \_CharT std::money\_get<\_CharT, \_InIter>::char\_type**

Public typedefs.

Definition at line 1376 of file locale\_facets\_nonio.h.

#### **5.599.2.2 template<typename \_CharT, typename \_InIter> typedef \_InIter std::money\_get<\_CharT, \_InIter>::iter\_type**

Public typedefs.

Definition at line 1377 of file locale\_facets\_nonio.h.

#### **5.599.2.3 template<typename \_CharT, typename \_InIter> typedef basic\_string<\_CharT> std::money\_get<\_CharT, \_InIter>::string\_type**

Public typedefs.

Definition at line 1378 of file locale\_facets\_nonio.h.

### **5.599.3 Constructor & Destructor Documentation**

#### **5.599.3.1 template<typename \_CharT, typename \_InIter> std::money\_get< \_CharT, \_InIter>::money\_get( size\_t \_\_refs = 0 ) [inline, explicit]**

Constructor performs initialization.

This is the constructor provided by the standard.

#### Parameters

*refs* Passed to the base facet class.

Definition at line 1392 of file `locale_facets_nonio.h`.

**5.599.3.2** `template<typename _CharT, typename _InIter> virtual  
std::money_get<_CharT, _InIter>::~~money_get( ) [inline,  
protected, virtual]`

Destructor.

Definition at line 1460 of file `locale_facets_nonio.h`.

#### 5.599.4 Member Function Documentation

**5.599.4.1** `template<typename _CharT, typename _InIter> _InIter  
std::money_get<_CharT, _InIter>::do_get( iter_type __s,  
iter_type __end, bool __intl, ios_base & __io, ios_base::iostate &  
__err, long double & __units ) const [protected, virtual]`

Read and parse a monetary value.

This function reads and parses characters representing a monetary value. This function is a hook for derived classes to change the value returned.

#### See also

[get\(\)](#) for details.

Definition at line 365 of file `locale_facets_nonio.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc>::c_str()`.

Referenced by `std::money_get<_CharT, _InIter>::get()`.

**5.599.4.2** `template<typename _CharT, typename _InIter> _InIter  
std::money_get<_CharT, _InIter>::do_get( iter_type __s,  
iter_type __end, bool __intl, ios_base & __io, ios_base::iostate &  
__err, string_type & __digits ) const [protected, virtual]`



Read and parse a monetary value.

This function reads and parses characters representing a monetary value. This function is a hook for derived classes to change the value returned.

#### See also

[get\(\)](#) for details.

Definition at line 378 of file locale\_facets\_nonio.tcc.

References `std::ios_base::_M_getloc()`, `std::basic_string<_CharT, _Traits, _Alloc>::resize()`, and `std::__ctype_abstract_base<_CharT>::widen()`.

**5.599.4.3** `template<typename _CharT, typename _InIter> iter_type  
std::money_get<_CharT, _InIter>::get ( iter_type __s, iter_type  
__end, bool __intl, ios_base & __io, ios_base::iostate & __err,  
string_type & __digits ) const [inline]`

Read and parse a monetary value.

This function reads characters from *s*, interprets them as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and returns the result in *digits*. For example, the string \$10.01 in a US locale would store 1001 in *digits*.

Any characters not part of a valid money amount are not consumed.

If a money value cannot be parsed from the input stream, sets `err=(err|io.failbit)`. If the stream is consumed before finishing parsing, sets `err=(err|io.failbit|io.eofbit)`.

This function works by returning the result of [do\\_get\(\)](#).

#### Parameters

- s* Start of characters to parse.
- end* End of characters to parse.
- intl* Parameter to use `_facet<moneypunct<CharT,intl>>`.
- io* Source of facets and io state.
- err* Error field to set if parsing fails.
- digits* Place to store result of parsing.

#### Returns

Iterator referencing first character beyond valid money amount.

Definition at line 1453 of file locale\_facets\_nonio.h.

References `std::money_get<_CharT, _InIter>::do_get()`.

**5.599.4.4** `template<typename _CharT, typename _InIter> iter_type  
std::money_get<_CharT, _InIter>::get ( iter_type __s, iter_type  
__end, bool __intl, ios_base & __io, ios_base::iostate & __err, long  
double & __units ) const [inline]`

Read and parse a monetary value.

This function reads characters from *s*, interprets them as a monetary value according to moneypunct and ctype facets retrieved from io.getloc(), and returns the result in *units* as an integral value `moneypunct::frac_digits()` \* the actual amount. For example, the string \$10.01 in a US locale would store 1001 in *units*.

Any characters not part of a valid money amount are not consumed.

If a money value cannot be parsed from the input stream, sets *err*=(*err*|io.failbit). If the stream is consumed before finishing parsing, sets *err*=(*err*|io.failbit|io.eofbit). *units* is unchanged if parsing fails.

This function works by returning the result of `do_get()`.

#### Parameters

- s* Start of characters to parse.
- end* End of characters to parse.
- intl* Parameter to use `facet<moneypunct<CharT,intl>>`.
- io* Source of facets and io state.
- err* Error field to set if parsing fails.
- units* Place to store result of parsing.

#### Returns

Iterator referencing first character beyond valid money amount.

Definition at line 1422 of file locale\_facets\_nonio.h.

References `std::money_get<_CharT, _InIter>::do_get()`.

### 5.599.5 Member Data Documentation

**5.599.5.1** `template<typename _CharT, typename _InIter> locale::id  
std::money_get<_CharT, _InIter>::id [static]`

Numpunct facet id.

Definition at line 1382 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following files:

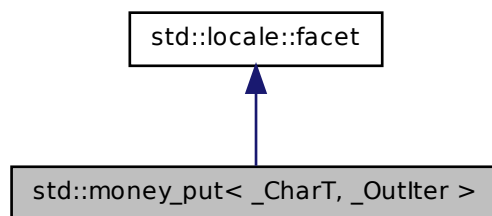
- [locale\\_facets\\_nonio.h](#)
- [locale\\_facets\\_nonio.tcc](#)

## 5.600 `std::money_put< _CharT, _OutIter >` Class Template Reference

Primary class template [money\\_put](#).

This facet encapsulates the code to format and output a monetary amount.

Inheritance diagram for `std::money_put< _CharT, _OutIter >`:



### Public Types

- typedef `_CharT` [char\\_type](#)
- typedef `_OutIter` [iter\\_type](#)
- typedef `basic_string< _CharT >` [string\\_type](#)

### Public Member Functions

- [money\\_put](#) (`size_t __refs=0`)
- [iter\\_type put](#) ([iter\\_type](#) \_\_s, `bool __intl`, `ios_base &__io`, [char\\_type](#) \_\_fill, `const string\_type &__digits`) `const`
- [iter\\_type put](#) ([iter\\_type](#) \_\_s, `bool __intl`, `ios_base &__io`, [char\\_type](#) \_\_fill, `long double __units`) `const`

### Static Public Attributes

- static `locale::id` `id`

### Protected Member Functions

- virtual `~money_put()`
- `template<bool _Intl>`  
`iter_type _M_insert (iter_type __s, ios_base &__io, char_type __fill, const`  
`string_type &__digits) const`
- virtual `iter_type do_put (iter_type __s, bool __intl, ios_base &__io, char_type`  
`__fill, const string_type &__digits) const`
- virtual `iter_type do_put (iter_type __s, bool __intl, ios_base &__io, char_type`  
`__fill, long double __units) const`

### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static `void _S_create_c_locale (__c_locale &__cloc, const char *__s, __c_`  
`locale __old=0)`
- static `void _S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static `const char * _S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

### Friends

- class `locale::_Impl`

#### 5.600.1 Detailed Description

`template<typename _CharT, typename _OutIter> class std::money_put< _`  
`CharT, _OutIter >`

Primary class template `money_put`.

This facet encapsulates the code to format and output a monetary amount. The `money_`  
`put` template uses protected virtual functions to provide the actual results. The public  
 accessors forward the call to the virtual functions. These virtual functions are hooks  
 for developers to implement the behavior they require from the `money_put` facet.

Definition at line 1521 of file `locale_facets_nonio.h`.

## 5.600.2 Member Typedef Documentation

**5.600.2.1** `template<typename _CharT, typename _OutIter > typedef _CharT  
std::money_put< _CharT, _OutIter >::char_type`

Public typedefs.

Definition at line 1526 of file locale\_facets\_nonio.h.

**5.600.2.2** `template<typename _CharT, typename _OutIter > typedef _OutIter  
std::money_put< _CharT, _OutIter >::iter_type`

Public typedefs.

Definition at line 1527 of file locale\_facets\_nonio.h.

**5.600.2.3** `template<typename _CharT, typename _OutIter > typedef  
basic_string<_CharT> std::money_put< _CharT, _OutIter  
>::string_type`

Public typedefs.

Definition at line 1528 of file locale\_facets\_nonio.h.

## 5.600.3 Constructor & Destructor Documentation

**5.600.3.1** `template<typename _CharT, typename _OutIter >  
std::money_put< _CharT, _OutIter >::money_put ( size_t __refs =  
0 ) [inline, explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

### Parameters

*refs* Passed to the base facet class.

Definition at line 1542 of file locale\_facets\_nonio.h.

**5.600.3.2** `template<typename _CharT, typename _OutIter > virtual  
std::money_put< _CharT, _OutIter >::~~money_put ( )  
[inline, protected, virtual]`

Destructor.

Definition at line 1592 of file locale\_facets\_nonio.h.

#### 5.600.4 Member Function Documentation

**5.600.4.1** `template<typename _CharT, typename _OutIter > _OutIter  
std::money_put< _CharT, _OutIter >::do_put ( iter_type __s, bool  
__intl, ios_base & __io, char_type __fill, long double __units )  
const [protected, virtual]`

Format and output a monetary value.

This function formats *units* as a monetary value according to moneypunct and ctype facets retrieved from io.getloc(), and writes the resulting characters to *s*. For example, the value 1001 in a US locale would write \$10.01 to *s*.

This function is a hook for derived classes to change the value returned.

#### See also

[put\(\)](#).

#### Parameters

*s* The stream to write to.

*intl* Parameter to use\_facet<moneypunct<CharT,intl> >.

*io* Source of facets and io state.

*fill* char\_type to use for padding.

*units* Place to store result of parsing.

#### Returns

Iterator after writing.

Definition at line 570 of file locale\_facets\_nonio.tcc.

References `std::ios_base::getloc()`, and `std::__ctype_abstract_base< _CharT >::widen()`.

Referenced by `std::money_put< _CharT, _OutIter >::put()`.

**5.600.4.2** `template<typename _CharT, typename _OutIter> _OutIter  
std::money_put<_CharT, _OutIter>::do_put ( iter_type __s, bool  
__intl, ios_base & __io, char_type __fill, const string_type &  
__digits ) const [protected, virtual]`

Format and output a monetary value.

This function formats *digits* as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and writes the resulting characters to *s*. For example, the string `1001` in a US locale would write `$10.01` to *s*.

This function is a hook for derived classes to change the value returned.

**See also**

[put\(\)](#).

**Parameters**

*s* The stream to write to.

*intl* Parameter to use `_facet<moneypunct<CharT,intl>>`.

*io* Source of facets and io state.

*fill* `char_type` to use for padding.

*units* Place to store result of parsing.

**Returns**

Iterator after writing.

Definition at line 608 of file `locale_facets_nonio.tcc`.

**5.600.4.3** `template<typename _CharT, typename _OutIter> iter_type  
std::money_put<_CharT, _OutIter>::put ( iter_type __s, bool  
__intl, ios_base & __io, char_type __fill, const string_type &  
__digits ) const [inline]`

Format and output a monetary value.

This function formats *digits* as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and writes the resulting characters to *s*. For example, the string `1001` in a US locale would write `$10.01` to *s*.

This function works by returning the result of [do\\_put\(\)](#).

**Parameters**

- s* The stream to write to.
- intl* Parameter to use\_facet<moneypunct<CharT,intl> >.
- io* Source of facets and io state.
- fill* char\_type to use for padding.
- units* Place to store result of parsing.

**Returns**

Iterator after writing.

Definition at line 1585 of file locale\_facets\_nonio.h.

References std::money\_put<\_CharT, \_OutIter>::do\_put().

```
5.600.4.4 template<typename _CharT, typename _OutIter> iter_type
std::money_put<_CharT, _OutIter>::put (iter_type __s, bool
__intl, ios_base & __io, char_type __fill, long double __units)
const [inline]
```

Format and output a monetary value.

This function formats *units* as a monetary value according to moneypunct and ctype facets retrieved from io.getloc(), and writes the resulting characters to *s*. For example, the value 1001 in a US locale would write \$10.01 to *s*.

This function works by returning the result of [do\\_put\(\)](#).

**Parameters**

- s* The stream to write to.
- intl* Parameter to use\_facet<moneypunct<CharT,intl> >.
- io* Source of facets and io state.
- fill* char\_type to use for padding.
- units* Place to store result of parsing.

**Returns**

Iterator after writing.

Definition at line 1562 of file locale\_facets\_nonio.h.

References std::money\_put<\_CharT, \_OutIter>::do\_put().



### 5.600.5 Member Data Documentation

#### 5.600.5.1 `template<typename _CharT, typename _OutIter> locale::id std::money_put<_CharT, _OutIter>::id [static]`

Numpunct facet id.

Definition at line 1532 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following files:

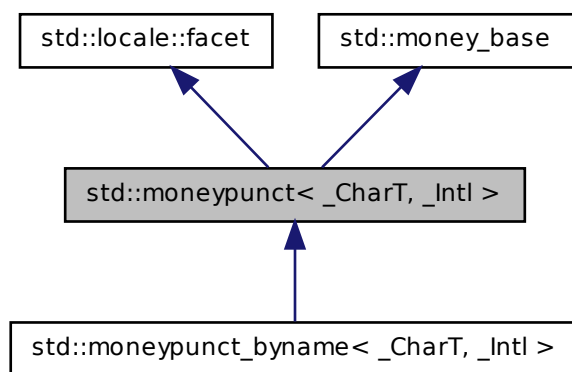
- [locale\\_facets\\_nonio.h](#)
- [locale\\_facets\\_nonio.tcc](#)

### 5.601 `std::moneypunct<_CharT, _Intl>` Class Template Reference

Primary class template `moneypunct`.

This facet encapsulates the punctuation, grouping and other formatting features of money amount string representations.

Inheritance diagram for `std::moneypunct<_CharT, _Intl>`:



### Public Types

- enum { `_S_minus`, `_S_zero`, `_S_end` }
- typedef `__moneypunct_cache<_CharT, _Intl >` `__cache_type`
- enum `part` {  
    `none`, `space`, `symbol`, `sign`,  
    `value` }
- typedef `_CharT` `char_type`
- typedef `basic_string<_CharT >` `string_type`

### Public Member Functions

- `moneypunct` (`size_t __refs=0`)
- `moneypunct` (`__cache_type *__cache`, `size_t __refs=0`)
- `moneypunct` (`_c_locale __cloc`, `const char *__s`, `size_t __refs=0`)
- `string_type curr_symbol` () const
- `char_type decimal_point` () const
- `int frac_digits` () const
- `string grouping` () const
- `string_type negative_sign` () const
- `string_type positive_sign` () const
- `char_type thousands_sep` () const
- pattern `pos_format` () const
- pattern `neg_format` () const

### Static Public Member Functions

- static pattern `_S_construct_pattern` (`char __precedes`, `char __space`, `char __-posn`) throw ()

### Static Public Attributes

- static const `char *` `_S_atoms`
- static const pattern `_S_default_pattern`
- static `locale::id` `id`
- static const bool `intl`

### Protected Member Functions

- virtual `~moneypunct()`
- template<>  
void `_M_initialize_moneypunct` (\_\_c\_locale, const char \*)
- void `_M_initialize_moneypunct` (\_\_c\_locale \_\_cloc=0, const char \*\_\_name=0)
- template<>  
void `_M_initialize_moneypunct` (\_\_c\_locale, const char \*)
- template<>  
void `_M_initialize_moneypunct` (\_\_c\_locale, const char \*)
- template<>  
void `_M_initialize_moneypunct` (\_\_c\_locale, const char \*)
- virtual `string_type do_curr_symbol()` const
- virtual `char_type do_decimal_point()` const
- virtual int `do_frac_digits()` const
- virtual `string do_grouping()` const
- virtual pattern `do_neg_format()` const
- virtual `string_type do_negative_sign()` const
- virtual pattern `do_pos_format()` const
- virtual `string_type do_positive_sign()` const
- virtual `char_type do_thousands_sep()` const

### Static Protected Member Functions

- static \_\_c\_locale `_S_clone_c_locale` (\_\_c\_locale &\_\_cloc) throw ()
- static void `_S_create_c_locale` (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void `_S_destroy_c_locale` (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale `_S_get_c_locale()`
- static const char \* `_S_get_c_name()` throw ()
- static \_\_c\_locale `_S_lc_ctype_c_locale` (\_\_c\_locale \_\_cloc, const char \*\_\_s)

### Friends

- class `locale::_Impl`

#### 5.601.1 Detailed Description

`template<typename _CharT, bool _Intl> class std::moneypunct<_CharT, _Intl>`

Primary class template `moneypunct`.

This facet encapsulates the punctuation, grouping and other formatting features of money amount string representations.

Definition at line 934 of file locale\_facets\_nonio.h.

### **5.601.2 Member Typedef Documentation**

#### **5.601.2.1 template<typename \_CharT, bool \_Intl> typedef \_CharT std::moneypunct<\_CharT, \_Intl>::char\_type**

Public typedefs.

Reimplemented in [std::moneypunct\\_byname<\\_CharT, \\_Intl>](#).

Definition at line 940 of file locale\_facets\_nonio.h.

#### **5.601.2.2 template<typename \_CharT, bool \_Intl> typedef basic\_string<\_CharT> std::moneypunct<\_CharT, \_Intl> >::string\_type**

Public typedefs.

Reimplemented in [std::moneypunct\\_byname<\\_CharT, \\_Intl>](#).

Definition at line 941 of file locale\_facets\_nonio.h.

### **5.601.3 Constructor & Destructor Documentation**

#### **5.601.3.1 template<typename \_CharT, bool \_Intl> std::moneypunct< \_CharT, \_Intl>::moneypunct ( size\_t \_\_refs = 0 ) [inline, explicit]**

Constructor performs initialization.

This is the constructor provided by the standard.

#### **Parameters**

*refs* Passed to the base facet class.

Definition at line 963 of file locale\_facets\_nonio.h.

**5.601.3.2** `template<typename _CharT, bool _Intl> std::moneypunct<  
_CharT, _Intl>::moneypunct ( __cache_type * __cache, size_t  
__refs = 0 ) [inline, explicit]`

Constructor performs initialization.

This is an internal constructor.

#### Parameters

*cache* Cache for optimization.

*refs* Passed to the base facet class.

Definition at line 976 of file locale\_facets\_nonio.h.

**5.601.3.3** `template<typename _CharT, bool _Intl> std::moneypunct<  
_CharT, _Intl>::moneypunct ( __c_locale __cloc, const char * __s,  
size_t __refs = 0 ) [inline, explicit]`

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

#### Parameters

*cloc* The C locale.

*s* The name of a locale.

*refs* Passed to the base facet class.

Definition at line 991 of file locale\_facets\_nonio.h.

**5.601.3.4** `template<typename _CharT, bool _Intl> virtual std::moneypunct<  
_CharT, _Intl>::~~moneypunct ( ) [protected, virtual]`

Destructor.

#### 5.601.4 Member Function Documentation

**5.601.4.1** `template<typename _CharT, bool _Intl> string_type  
std::moneypunct<_CharT, _Intl>::curr_symbol ( ) const  
[inline]`

Return currency symbol string.

This function returns a `string_type` to use as a currency symbol. It does so by returning returning `moneypunct<char_type>::do_curr_symbol()`.

##### Returns

*string\_type* representing a currency symbol.

Definition at line 1061 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl>::do_curr_symbol()`.

**5.601.4.2** `template<typename _CharT, bool _Intl> char_type  
std::moneypunct<_CharT, _Intl>::decimal_point ( ) const  
[inline]`

Return decimal point character.

This function returns a `char_type` to use as a decimal point. It does so by returning returning `moneypunct<char_type>::do_decimal_point()`.

##### Returns

*char\_type* representing a decimal point.

Definition at line 1005 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl>::do_decimal_point()`.

**5.601.4.3** `template<typename _CharT, bool _Intl> virtual string_type  
std::moneypunct<_CharT, _Intl>::do_curr_symbol ( ) const  
[inline, protected, virtual]`

Return currency symbol string.

This function returns a `string_type` to use as a currency symbol. This function is a hook for derived classes to change the value returned.

**See also**

[`curr\_symbol\(\)`](#) for details.

**Returns**

*string\_type* representing a currency symbol.

Definition at line 1207 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct<_CharT, _Intl>::curr_symbol()`.

**5.601.4.4** `template<typename _CharT, bool _Intl> virtual char_type  
std::moneypunct<_CharT, _Intl>::do_decimal_point ( ) const  
[inline, protected, virtual]`

Return decimal point character.

Returns a `char_type` to use as a decimal point. This function is a hook for derived classes to change the value returned.

**Returns**

*char\_type* representing a decimal point.

Definition at line 1169 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct<_CharT, _Intl>::decimal_point()`.

**5.601.4.5** `template<typename _CharT, bool _Intl> virtual int  
std::moneypunct<_CharT, _Intl>::do_frac_digits ( ) const  
[inline, protected, virtual]`

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. This function is a hook for derived classes to change the value returned.

**See also**

[`frac\_digits\(\)`](#) for details.

**Returns**

Number of digits in amount fraction.

Definition at line 1247 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct<_CharT, _Intl>::frac_digits()`.

**5.601.4.6** `template<typename _CharT, bool _Intl> virtual string  
std::moneypunct<_CharT, _Intl>::do_grouping ( ) const  
[inline, protected, virtual]`

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

**See also**

[grouping\(\)](#) for details.

**Returns**

String representing grouping specification.

Definition at line 1194 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct<_CharT, _Intl>::grouping()`.

**5.601.4.7** `template<typename _CharT, bool _Intl> virtual pattern  
std::moneypunct<_CharT, _Intl>::do_neg_format ( ) const  
[inline, protected, virtual]`

Return pattern for money values.

This function returns a pattern describing the formatting of a negative valued money amount. This function is a hook for derived classes to change the value returned.

**See also**

[neg\\_format\(\)](#) for details.

**Returns**

Pattern for money values.

Definition at line 1275 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct<_CharT, _Intl>::neg_format()`.



**5.601.4.8** `template<typename _CharT, bool _Intl> virtual string_type  
std::moneypunct<_CharT, _Intl>::do_negative_sign ( ) const  
[inline, protected, virtual]`

Return negative sign string.

This function returns a `string_type` to use as a sign for negative amounts. This function is a hook for derived classes to change the value returned.

**See also**

[negative\\_sign\(\)](#) for details.

**Returns**

*string\_type* representing a negative sign.

Definition at line 1233 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct<_CharT, _Intl>::negative_sign()`.

**5.601.4.9** `template<typename _CharT, bool _Intl> virtual pattern  
std::moneypunct<_CharT, _Intl>::do_pos_format ( ) const  
[inline, protected, virtual]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive valued money amount. This function is a hook for derived classes to change the value returned.

**See also**

[pos\\_format\(\)](#) for details.

**Returns**

Pattern for money values.

Definition at line 1261 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct<_CharT, _Intl>::pos_format()`.

**5.601.4.10** `template<typename _CharT, bool _Intl> virtual string_type  
std::moneypunct<_CharT, _Intl>::do_positive_sign ( ) const  
[inline, protected, virtual]`

Return positive sign string.

This function returns a `string_type` to use as a sign for positive amounts. This function is a hook for derived classes to change the value returned.

**See also**

[`positive\_sign\(\)`](#) for details.

**Returns**

*string\_type* representing a positive sign.

Definition at line 1220 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct<_CharT, _Intl>::positive_sign()`.

**5.601.4.11** `template<typename _CharT, bool _Intl> virtual char_type  
std::moneypunct<_CharT, _Intl>::do_thousands_sep ( ) const  
[inline, protected, virtual]`

Return thousands separator character.

Returns a `char_type` to use as a thousands separator. This function is a hook for derived classes to change the value returned.

**Returns**

*char\_type* representing a thousands separator.

Definition at line 1181 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct<_CharT, _Intl>::thousands_sep()`.

**5.601.4.12** `template<typename _CharT, bool _Intl> int std::moneypunct<  
_CharT, _Intl>::frac_digits ( ) const [inline]`

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. It does so by returning `moneypunct<char_type>::do_frac_digits()`.

The fractional part of a money amount is optional. But if it is present, there must be `frac_digits()` digits.

**Returns**

Number of digits in amount fraction.

Definition at line 1111 of file locale\_facets\_nonio.h.

References `std::moneypunct< _CharT, _Intl >::do_frac_digits()`.

**5.601.4.13 template<typename \_CharT, bool \_Intl> string std::moneypunct< \_CharT, \_Intl >::grouping ( ) const [inline]**

Return grouping specification.

This function returns a string representing groupings for the integer part of an amount. Groupings indicate where thousands separators should be inserted.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the `grouping()` returns `32` and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was `32`, this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling `moneypunct<char_type>::do_grouping()`.

**Returns**

string representing grouping specification.

Definition at line 1048 of file locale\_facets\_nonio.h.

References `std::moneypunct< _CharT, _Intl >::do_grouping()`.

**5.601.4.14 template<typename \_CharT, bool \_Intl> pattern std::moneypunct< \_CharT, \_Intl >::neg\_format ( ) const [inline]**

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning `moneypunct<char_type>::do_pos_format()` or `moneypunct<char_type>::do_neg_format()`.

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of `curr_symbol()` may be present. The sign field indicates that the value of `positive_sign()` or `negative_sign()` must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and `pos_format()` pattern {symbol,sign,value,none}, `curr_symbol() == '$'` `positive_sign() == '+'`, and value 10.01, and options set to force the symbol, the corresponding string is \$+10.01.

### Returns

Pattern for money values.

Definition at line 1151 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl>::do_neg_format()`.

**5.601.4.15** `template<typename _CharT, bool _Intl> string_type  
std::moneypunct<_CharT, _Intl>::negative_sign ( ) const  
[inline]`

Return negative sign string.

This function returns a `string_type` to use as a sign for negative amounts. It does so by returning `moneypunct<char_type>::do_negative_sign()`.

If the return value contains more than one character, the first character appears in the position indicated by `neg_format()` and the remainder appear at the end of the formatted string.

### Returns

*string\_type* representing a negative sign.

Definition at line 1095 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl>::do_negative_sign()`.

**5.601.4.16** `template<typename _CharT, bool _Intl> pattern  
std::moneypunct<_CharT, _Intl>::pos_format ( ) const  
[inline]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning `moneypunct<char_type>::do_pos_format()` or `moneypunct<char_type>::do_neg_format()`.

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of `curr_symbol()` may be present. The sign field indicates that the value of `positive_sign()` or `negative_sign()` must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and `pos_format()` pattern {symbol,sign,value,none}, `curr_symbol() == '$'` `positive_sign() == '+'`, and value 10.01, and options set to force the symbol, the corresponding string is \$+10.01.

#### Returns

Pattern for money values.

Definition at line 1147 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl>::do_pos_format()`.

**5.601.4.17** `template<typename _CharT, bool _Intl> string_type  
std::moneypunct<_CharT, _Intl>::positive_sign ( ) const  
[inline]`

Return positive sign string.

This function returns a `string_type` to use as a sign for positive amounts. It does so by returning `moneypunct<char_type>::do_positive_sign()`.

If the return value contains more than one character, the first character appears in the position indicated by `pos_format()` and the remainder appear at the end of the formatted string.

**Returns**

*string\_type* representing a positive sign.

Definition at line 1078 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl>::do_positive_sign()`.

**5.601.4.18** `template<typename _CharT, bool _Intl> char_type  
std::moneypunct<_CharT, _Intl>::thousands_sep ( ) const  
[inline]`

Return thousands separator character.

This function returns a `char_type` to use as a thousands separator. It does so by returning returning `moneypunct<char_type>::do_thousands_sep()`.

**Returns**

`char_type` representing a thousands separator.

Definition at line 1018 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl>::do_thousands_sep()`.

**5.601.5 Member Data Documentation**

**5.601.5.1** `template<typename _CharT, bool _Intl> locale::id  
std::moneypunct<_CharT, _Intl>::id [static]`

Numpunct facet id.

Definition at line 953 of file `locale_facets_nonio.h`.

**5.601.5.2** `template<typename _CharT, bool _Intl> const bool  
std::moneypunct<_CharT, _Intl>::intl [static]`

This value is provided by the standard, but no reason for its /// existence.

Reimplemented in `std::moneypunct_byname<_CharT, _Intl>`.

Definition at line 951 of file `locale_facets_nonio.h`.

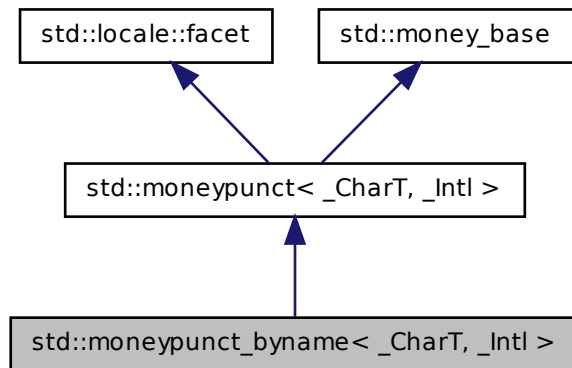
The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

## 5.602 `std::moneypunct_byname<_CharT, _Intl>` Class Template Reference

class [moneypunct\\_byname](#) [22.2.6.4].

Inheritance diagram for `std::moneypunct_byname<_CharT, _Intl>`:



### Public Types

- enum { `_S_minus`, `_S_zero`, `_S_end` }
- typedef `__moneypunct_cache<_CharT, _Intl>` `__cache_type`
- typedef `_CharT` `char_type`
- enum **part** {  
    `none`, `space`, `symbol`, `sign`,  
    `value` }
- typedef `basic_string<_CharT>` `string_type`

### Public Member Functions

- `moneypunct_byname` (`const char *__s`, `size_t __refs=0`)
- `string_type curr_symbol` () `const`

- `char_type decimal_point () const`
- `int frac_digits () const`
- `string grouping () const`
- `string_type negative_sign () const`
- `string_type positive_sign () const`
- `char_type thousands_sep () const`

- `pattern pos_format () const`
- `pattern neg_format () const`

### Static Public Member Functions

- `static pattern _S_construct_pattern (char __precedes, char __space, char __posn) throw ()`

### Static Public Attributes

- `static const char * _S_atoms`
- `static const pattern _S_default_pattern`
- `static locale::id id`
- `static const bool intl`

### Protected Member Functions

- `template<>`  
`void _M_initialize_moneypunct (__c_locale, const char *)`
- `void _M_initialize_moneypunct (__c_locale __cloc=0, const char *__name=0)`
- `template<>`  
`void _M_initialize_moneypunct (__c_locale, const char *)`
- `template<>`  
`void _M_initialize_moneypunct (__c_locale, const char *)`
- `template<>`  
`void _M_initialize_moneypunct (__c_locale, const char *)`
- `virtual string_type do_curr_symbol () const`
- `virtual char_type do_decimal_point () const`
- `virtual int do_frac_digits () const`
- `virtual string do_grouping () const`
- `virtual pattern do_neg_format () const`
- `virtual string_type do_negative_sign () const`
- `virtual pattern do_pos_format () const`
- `virtual string_type do_positive_sign () const`
- `virtual char_type do_thousands_sep () const`



### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static const char \* `_S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

### Friends

- class `locale::_Impl`

#### 5.602.1 Detailed Description

`template<typename _CharT, bool _Intl> class std::moneypunct_byname< _CharT, _Intl >`

class [moneypunct\\_byname](#) [22.2.6.4].

Definition at line 1324 of file `locale_facets_nonio.h`.

#### 5.602.2 Member Typedef Documentation

**5.602.2.1** `template<typename _CharT, bool _Intl> typedef _CharT  
std::moneypunct_byname< _CharT, _Intl >::char_type`

Public typedefs.

Reimplemented from [std::moneypunct< \\_CharT, \\_Intl >](#).

Definition at line 1327 of file `locale_facets_nonio.h`.

**5.602.2.2** `template<typename _CharT, bool _Intl> typedef  
basic_string<_CharT> std::moneypunct_byname< _CharT, _Intl  
>::string_type`

Public typedefs.

Reimplemented from [std::moneypunct< \\_CharT, \\_Intl >](#).

Definition at line 1328 of file `locale_facets_nonio.h`.

### 5.602.3 Member Function Documentation

**5.602.3.1** `template<typename _CharT, bool _Intl> string_type  
std::moneypunct< _CharT, _Intl >::curr_symbol ( ) const  
[inline, inherited]`

Return currency symbol string.

This function returns a `string_type` to use as a currency symbol. It does so by returning returning `moneypunct<char_type>::do_curr_symbol()`.

#### Returns

*string\_type* representing a currency symbol.

Definition at line 1061 of file `locale_facets_nonio.h`.

References `std::moneypunct< _CharT, _Intl >::do_curr_symbol()`.

**5.602.3.2** `template<typename _CharT, bool _Intl> char_type  
std::moneypunct< _CharT, _Intl >::decimal_point ( ) const  
[inline, inherited]`

Return decimal point character.

This function returns a `char_type` to use as a decimal point. It does so by returning returning `moneypunct<char_type>::do_decimal_point()`.

#### Returns

*char\_type* representing a decimal point.

Definition at line 1005 of file `locale_facets_nonio.h`.

References `std::moneypunct< _CharT, _Intl >::do_decimal_point()`.

**5.602.3.3** `template<typename _CharT, bool _Intl> virtual string_type  
std::moneypunct< _CharT, _Intl >::do_curr_symbol ( ) const  
[inline, protected, virtual, inherited]`

Return currency symbol string.

This function returns a `string_type` to use as a currency symbol. This function is a hook for derived classes to change the value returned.

**See also**

[`curr\_symbol\(\)`](#) for details.

**Returns**

*string\_type* representing a currency symbol.

Definition at line 1207 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct<_CharT, _Intl>::curr_symbol()`.

**5.602.3.4** `template<typename _CharT, bool _Intl> virtual char_type  
std::moneypunct<_CharT, _Intl>::do_decimal_point ( ) const  
[inline, protected, virtual, inherited]`

Return decimal point character.

Returns a `char_type` to use as a decimal point. This function is a hook for derived classes to change the value returned.

**Returns**

*char\_type* representing a decimal point.

Definition at line 1169 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct<_CharT, _Intl>::decimal_point()`.

**5.602.3.5** `template<typename _CharT, bool _Intl> virtual int  
std::moneypunct<_CharT, _Intl>::do_frac_digits ( ) const  
[inline, protected, virtual, inherited]`

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. This function is a hook for derived classes to change the value returned.

**See also**

[`frac\_digits\(\)`](#) for details.

**Returns**

Number of digits in amount fraction.

## 5.602 `std::moneypunct_byname<_CharT, _Intl>` Class Template Reference 2977

---

Definition at line 1247 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct<_CharT, _Intl>::frac_digits()`.

**5.602.3.6** `template<typename _CharT, bool _Intl> virtual string  
std::moneypunct<_CharT, _Intl>::do_grouping ( ) const  
[inline, protected, virtual, inherited]`

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

### See also

[`grouping\(\)`](#) for details.

### Returns

String representing grouping specification.

Definition at line 1194 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct<_CharT, _Intl>::grouping()`.

**5.602.3.7** `template<typename _CharT, bool _Intl> virtual pattern  
std::moneypunct<_CharT, _Intl>::do_neg_format ( ) const  
[inline, protected, virtual, inherited]`

Return pattern for money values.

This function returns a pattern describing the formatting of a negative valued money amount. This function is a hook for derived classes to change the value returned.

### See also

[`neg\_format\(\)`](#) for details.

### Returns

Pattern for money values.

Definition at line 1275 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct<_CharT, _Intl>::neg_format()`.

**5.602.3.8** `template<typename _CharT, bool _Intl> virtual string_type  
std::moneypunct< _CharT, _Intl >::do_negative_sign ( ) const  
[inline, protected, virtual, inherited]`

Return negative sign string.

This function returns a `string_type` to use as a sign for negative amounts. This function is a hook for derived classes to change the value returned.

**See also**

[negative\\_sign\(\)](#) for details.

**Returns**

*string\_type* representing a negative sign.

Definition at line 1233 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct< _CharT, _Intl >::negative_sign()`.

**5.602.3.9** `template<typename _CharT, bool _Intl> virtual pattern  
std::moneypunct< _CharT, _Intl >::do_pos_format ( ) const  
[inline, protected, virtual, inherited]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive valued money amount. This function is a hook for derived classes to change the value returned.

**See also**

[pos\\_format\(\)](#) for details.

**Returns**

Pattern for money values.

Definition at line 1261 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct< _CharT, _Intl >::pos_format()`.

**5.602.3.10** `template<typename _CharT, bool _Intl> virtual string_type  
std::moneypunct< _CharT, _Intl >::do_positive_sign ( ) const  
[inline, protected, virtual, inherited]`

Return positive sign string.

This function returns a `string_type` to use as a sign for positive amounts. This function is a hook for derived classes to change the value returned.

#### See also

[`positive\_sign\(\)`](#) for details.

#### Returns

*string\_type* representing a positive sign.

Definition at line 1220 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct<_CharT, _Intl>::positive_sign()`.

**5.602.3.11** `template<typename _CharT, bool _Intl> virtual char_type  
std::moneypunct<_CharT, _Intl>::do_thousands_sep ( ) const  
[inline, protected, virtual, inherited]`

Return thousands separator character.

Returns a `char_type` to use as a thousands separator. This function is a hook for derived classes to change the value returned.

#### Returns

*char\_type* representing a thousands separator.

Definition at line 1181 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct<_CharT, _Intl>::thousands_sep()`.

**5.602.3.12** `template<typename _CharT, bool _Intl> int std::moneypunct<  
_CharT, _Intl>::frac_digits ( ) const [inline, inherited]`

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. It does so by returning `moneypunct<char_type>::do_frac_digits()`.

The fractional part of a money amount is optional. But if it is present, there must be `frac_digits()` digits.

**Returns**

Number of digits in amount fraction.

Definition at line 1111 of file locale\_facets\_nonio.h.

References `std::moneypunct<_CharT, _Intl>::do_frac_digits()`.

**5.602.3.13** `template<typename _CharT, bool _Intl> string std::moneypunct<_CharT, _Intl>::grouping( ) const [inline, inherited]`

Return grouping specification.

This function returns a string representing groupings for the integer part of an amount. Groupings indicate where thousands separators should be inserted.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the `grouping()` returns `32` and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was `32`, this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling `moneypunct<char_type>::do_grouping()`.

**Returns**

string representing grouping specification.

Definition at line 1048 of file locale\_facets\_nonio.h.

References `std::moneypunct<_CharT, _Intl>::do_grouping()`.

**5.602.3.14** `template<typename _CharT, bool _Intl> pattern std::moneypunct<_CharT, _Intl>::neg_format( ) const [inline, inherited]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning `moneypunct<char_type>::do_pos_format()` or `moneypunct<char_type>::do_neg_format()`.

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of `curr_symbol()` may be present. The sign field indicates that the value of `positive_sign()` or `negative_sign()` must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and `pos_format()` pattern {symbol,sign,value,none}, `curr_symbol() == '$'` `positive_sign() == '+'`, and value 10.01, and options set to force the symbol, the corresponding string is `$+10.01`.

### Returns

Pattern for money values.

Definition at line 1151 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl>::do_neg_format()`.

**5.602.3.15** `template<typename _CharT, bool _Intl> string_type  
std::moneypunct<_CharT, _Intl>::negative_sign ( ) const  
[inline, inherited]`

Return negative sign string.

This function returns a `string_type` to use as a sign for negative amounts. It does so by returning `moneypunct<char_type>::do_negative_sign()`.

If the return value contains more than one character, the first character appears in the position indicated by `neg_format()` and the remainder appear at the end of the formatted string.

### Returns

*string\_type* representing a negative sign.

Definition at line 1095 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl>::do_negative_sign()`.



**5.602.3.16** `template<typename _CharT, bool _Intl> pattern  
std::moneypunct<_CharT, _Intl>::pos_format ( ) const  
[inline, inherited]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning `moneypunct<char_type>::do_pos_format()` or `moneypunct<char_type>::do_neg_format()`.

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of `curr_symbol()` may be present. The sign field indicates that the value of `positive_sign()` or `negative_sign()` must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and `pos_format()` pattern {symbol,sign,value,none}, `curr_symbol() == '$'` `positive_sign() == '+'`, and value 10.01, and options set to force the symbol, the corresponding string is \$+10.01.

### Returns

Pattern for money values.

Definition at line 1147 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl>::do_pos_format()`.

**5.602.3.17** `template<typename _CharT, bool _Intl> string_type  
std::moneypunct<_CharT, _Intl>::positive_sign ( ) const  
[inline, inherited]`

Return positive sign string.

This function returns a `string_type` to use as a sign for positive amounts. It does so by returning `moneypunct<char_type>::do_positive_sign()`.

If the return value contains more than one character, the first character appears in the position indicated by `pos_format()` and the remainder appear at the end of the formatted string.

**Returns**

*string\_type* representing a positive sign.

Definition at line 1078 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl>::do_positive_sign()`.

**5.602.3.18** `template<typename _CharT, bool _Intl> char_type  
std::moneypunct<_CharT, _Intl>::thousands_sep ( ) const  
[inline, inherited]`

Return thousands separator character.

This function returns a `char_type` to use as a thousands separator. It does so by returning  
returning `moneypunct<char_type>::do_thousands_sep()`.

**Returns**

`char_type` representing a thousands separator.

Definition at line 1018 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl>::do_thousands_sep()`.

**5.602.4 Member Data Documentation**

**5.602.4.1** `template<typename _CharT, bool _Intl> locale::id  
std::moneypunct<_CharT, _Intl>::id [static, inherited]`

Numpunct facet id.

Definition at line 953 of file `locale_facets_nonio.h`.

**5.602.4.2** `template<typename _CharT, bool _Intl> const bool  
std::moneypunct_byname<_CharT, _Intl>::intl [static]`

This value is provided by the standard, but no reason for its /// existence.

Reimplemented from `std::moneypunct<_CharT, _Intl>`.

Definition at line 1330 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

## 5.603 `std::move_iterator< _Iterator >` Class Template Reference

### Public Types

- typedef `__traits_type::difference_type` **difference\_type**
- typedef `__traits_type::iterator_category` **iterator\_category**
- typedef `_Iterator` **iterator\_type**
- typedef `_Iterator` **pointer**
- typedef `value_type &&` **reference**
- typedef `__traits_type::value_type` **value\_type**

### Public Member Functions

- **move\_iterator** (`iterator_type __i`)
- `template<typename _Iter >`  
**move\_iterator** (`const move\_iterator< _Iter > &__i`)
- `iterator_type` **base** () const
- reference **operator\*** () const
- [move\\_iterator](#) **operator+** (`difference_type __n`) const
- [move\\_iterator](#) **operator++** (`int`)
- [move\\_iterator](#) & **operator++** ()
- [move\\_iterator](#) & **operator+=** (`difference_type __n`)
- [move\\_iterator](#) **operator-** (`difference_type __n`) const
- [move\\_iterator](#) & **operator--** ()
- [move\\_iterator](#) **operator--** (`int`)
- [move\\_iterator](#) & **operator-=** (`difference_type __n`)
- pointer **operator->** () const
- reference **operator[]** (`difference_type __n`) const

### Protected Types

- typedef `iterator_traits< _Iterator > __traits_type`

### Protected Attributes

- `_Iterator` **\_M\_current**

### 5.603.1 Detailed Description

**template<typename \_Iterator> class std::move\_iterator< \_Iterator >**

Class template [move\\_iterator](#) is an iterator adapter with the same behavior as the underlying iterator except that its dereference operator implicitly converts the value returned by the underlying iterator's dereference operator to an rvalue reference. Some generic algorithms can be called with move iterators to replace copying with moving.

Definition at line 924 of file [stl\\_iterator.h](#).

The documentation for this class was generated from the following file:

- [stl\\_iterator.h](#)

## 5.604 std::multimap< \_Key, \_Tp, \_Compare, \_Alloc > Class Template Reference

A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.

### Public Types

- typedef \_Alloc **allocator\_type**
- typedef \_Rep\_type::const\_iterator **const\_iterator**
- typedef \_Pair\_alloc\_type::const\_pointer **const\_pointer**
- typedef \_Pair\_alloc\_type::const\_reference **const\_reference**
- typedef [\\_Rep\\_type::const\\_reverse\\_iterator](#) **const\_reverse\_iterator**
- typedef \_Rep\_type::difference\_type **difference\_type**
- typedef \_Rep\_type::iterator **iterator**
- typedef \_Compare **key\_compare**
- typedef \_Key **key\_type**
- typedef \_Tp **mapped\_type**
- typedef \_Pair\_alloc\_type::pointer **pointer**
- typedef \_Pair\_alloc\_type::reference **reference**
- typedef [\\_Rep\\_type::reverse\\_iterator](#) **reverse\_iterator**
- typedef \_Rep\_type::size\_type **size\_type**
- typedef [std::pair](#)< const \_Key, \_Tp > **value\_type**

### Public Member Functions

- [multimap](#) ()
- [multimap](#) (const \_Compare &\_\_comp, const allocator\_type &\_\_a=allocator\_type())

- `multimap` (`multimap` &&\_\_x)
- `multimap` (`initializer_list`< `value_type` > \_\_l, const `_Compare` &\_\_comp=`_Compare()`, const `allocator_type` &\_\_a=`allocator_type()`)
- `multimap` (const `multimap` &\_\_x)
- `template`<typename `_InputIterator` >  
  `multimap` (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- `template`<typename `_InputIterator` >  
  `multimap` (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Compare` &\_\_comp, const `allocator_type` &\_\_a=`allocator_type()`)
- iterator `begin` ()
- const\_iterator `begin` () const
- const\_iterator `cbegin` () const
- const\_iterator `end` () const
- void `clear` ()
- size\_type `count` (const key\_type &\_\_x) const
- const\_reverse\_iterator `crbegin` () const
- const\_reverse\_iterator `crend` () const
- bool `empty` () const
- iterator `end` ()
- const\_iterator `end` () const
- `std::pair`< const\_iterator, const\_iterator > `equal_range` (const key\_type &\_\_x) const
- `std::pair`< iterator, iterator > `equal_range` (const key\_type &\_\_x)
- iterator `erase` (const\_iterator \_\_first, const\_iterator \_\_last)
- iterator `erase` (const\_iterator \_\_position)
- size\_type `erase` (const key\_type &\_\_x)
- iterator `find` (const key\_type &\_\_x)
- const\_iterator `find` (const key\_type &\_\_x) const
- allocator\_type `get_allocator` () const
- void `insert` (`initializer_list`< `value_type` > \_\_l)
- `template`<typename `_InputIterator` >  
  void `insert` (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- iterator `insert` (const `value_type` &\_\_x)
- `template`<typename `_Pair`, typename = typename `std::enable_if`<`std::is_convertible`<`_Pair`, `value_type`>::value>::type>  
  iterator `insert` (`_Pair` &&\_\_x)
- iterator `insert` (const\_iterator \_\_position, const `value_type` &\_\_x)
- `template`<typename `_Pair`, typename = typename `std::enable_if`<`std::is_convertible`<`_Pair`, `value_type`>::value>::type>  
  iterator `insert` (const\_iterator \_\_position, `_Pair` &&\_\_x)
- key\_compare `key_comp` () const
- const\_iterator `lower_bound` (const key\_type &\_\_x) const
- iterator `lower_bound` (const key\_type &\_\_x)

- `size_type max_size () const`
- `multimap & operator= (const multimap &__x)`
- `multimap & operator= (multimap &&__x)`
- `multimap & operator= (initializer_list< value_type > __l)`
- `const_reverse_iterator rbegin () const`
- `reverse_iterator rbegin ()`
- `reverse_iterator rend ()`
- `const_reverse_iterator rend () const`
- `size_type size () const`
- `void swap (multimap &__x)`
- `iterator upper_bound (const key_type &__x)`
- `const_iterator upper_bound (const key_type &__x) const`
- `value_compare value_comp () const`

#### Friends

- `template<typename _K1, typename _T1, typename _C1, typename _A1 >  
bool operator< (const multimap< _K1, _T1, _C1, _A1 > &, const multimap<  
_K1, _T1, _C1, _A1 > &)`
- `template<typename _K1, typename _T1, typename _C1, typename _A1 >  
bool operator== (const multimap< _K1, _T1, _C1, _A1 > &, const multimap<  
_K1, _T1, _C1, _A1 > &)`

#### 5.604.1 Detailed Description

`template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> class std::multimap< _Key, _Tp, _Compare, _Alloc >`

A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time. Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using equivalent keys). For a `multimap<Key, T>` the `key_type` is `Key`, the `mapped_type` is `T`, and the `value_type` is `std::pair<const Key,T>`.

Multimaps support bidirectional iterators.

The private tree data is declared exactly the same way for `map` and `multimap`; the distinction is made entirely in how the tree functions are called (`*_unique` versus `*_equal`, same as the standard).

Definition at line 88 of file `stl_multimap.h`.

## 5.604.2 Constructor & Destructor Documentation

**5.604.2.1** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> std::multimap<_Key, _Tp, _Compare, _Alloc  
>::multimap ( ) [inline]`

Default constructor creates no elements.

Definition at line 150 of file `stl_multimap.h`.

**5.604.2.2** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> std::multimap<_Key, _Tp, _Compare, _Alloc  
>::multimap ( const _Compare & __comp, const allocator_type &  
__a = allocator_type() ) [inline, explicit]`

Creates a multimap with no elements.

### Parameters

*comp* A comparison object.

*a* An allocator object.

Definition at line 159 of file `stl_multimap.h`.

**5.604.2.3** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> std::multimap<_Key, _Tp, _Compare, _Alloc  
>::multimap ( const multimap<_Key, _Tp, _Compare, _Alloc> &  
__x ) [inline]`

Multimap copy constructor.

### Parameters

*x* A multimap of identical element and allocator types.

The newly-created multimap uses a copy of the allocation object used by *x*.

Definition at line 170 of file `stl_multimap.h`.

**5.604.2.4** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> std::multimap< _Key, _Tp, _Compare, _Alloc  
>::multimap ( multimap< _Key, _Tp, _Compare, _Alloc > && __x  
) [inline]`

Multimap move constructor.

#### Parameters

*x* A multimap of identical element and allocator types.

The newly-created multimap contains the exact contents of *x*. The contents of *x* are a valid, but unspecified multimap.

Definition at line 181 of file `std_multimap.h`.

**5.604.2.5** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> std::multimap< _Key, _Tp, _Compare, _Alloc  
>::multimap ( initializer_list< value_type > __l, const _Compare  
& __comp = _Compare(), const allocator_type & __a =  
allocator_type() ) [inline]`

Builds a multimap from an [initializer\\_list](#).

#### Parameters

*l* An [initializer\\_list](#).

*comp* A comparison functor.

*a* An allocator object.

Create a multimap consisting of copies of the elements from the [initializer\\_list](#). This is linear in *N* if the list is already sorted, and *N*log*N* otherwise (where *N* is `__l.size()`).

Definition at line 194 of file `std_multimap.h`.

**5.604.2.6** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> template<typename _InputIterator >  
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (   
_InputIterator __first, _InputIterator __last ) [inline]`



Builds a multimap from a range.

**Parameters**

*first* An input iterator.

*last* An input iterator.

Create a multimap consisting of copies of the elements from [first,last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(first,last)).

Definition at line 211 of file stl\_multimap.h.

```
5.604.2.7 template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> template<typename _InputIterator >
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
_InputIterator __first, _InputIterator __last, const _Compare &
__comp, const allocator_type & __a = allocator_type())
[inline]
```

Builds a multimap from a range.

**Parameters**

*first* An input iterator.

*last* An input iterator.

*comp* A comparison functor.

*a* An allocator object.

Create a multimap consisting of copies of the elements from [first,last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(first,last)).

Definition at line 227 of file stl\_multimap.h.

**5.604.3 Member Function Documentation**

```
5.604.3.1 template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> iterator std::multimap< _Key, _Tp, _Compare,
_Alloc >::begin () [inline]
```

Returns a read/write iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 306 of file stl\_multimap.h.

**5.604.3.2** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> const_iterator std::multimap< _Key, _Tp,  
_Compare, _Alloc >::begin ( ) const [inline]`

Returns a read-only (constant) iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 315 of file stl\_multimap.h.

**5.604.3.3** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> const_iterator std::multimap< _Key, _Tp,  
_Compare, _Alloc >::cbegin ( ) const [inline]`

Returns a read-only (constant) iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 379 of file stl\_multimap.h.

**5.604.3.4** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> const_iterator std::multimap< _Key, _Tp,  
_Compare, _Alloc >::cend ( ) const [inline]`

Returns a read-only (constant) iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 388 of file stl\_multimap.h.

**5.604.3.5** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> void std::multimap< _Key, _Tp, _Compare, _Alloc  
>::clear ( ) [inline]`

Erases all elements in a multimap. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 628 of file stl\_multimap.h.

Referenced by `std::multimap< _Key, _Tp, _Compare, _Alloc >::operator=()`.

**5.604.3.6** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> size_type std::multimap< _Key, _Tp, _Compare,  
_Alloc >::count ( const key_type & __x ) const [inline]`

Finds the number of elements with given key.

**Parameters**

*x* Key of (key, value) pairs to be located.

**Returns**

Number of elements with specified key.

Definition at line 685 of file `stl_multimap.h`.

**5.604.3.7** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> const_reverse_iterator std::multimap< _Key, _Tp,  
_Compare, _Alloc >::crbegin ( ) const [inline]`

Returns a read-only (constant) reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 397 of file `stl_multimap.h`.

**5.604.3.8** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> const_reverse_iterator std::multimap< _Key, _Tp,  
_Compare, _Alloc >::crend ( ) const [inline]`

Returns a read-only (constant) reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 406 of file `stl_multimap.h`.

**5.604.3.9** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> bool std::multimap< _Key, _Tp, _Compare, _Alloc  
>::empty ( ) const [inline]`

Returns true if the multimap is empty.

Definition at line 413 of file `stl_multimap.h`.

**5.604.3.10** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> const_iterator std::multimap< _Key, _Tp,  
_Compare, _Alloc >::end ( ) const [inline]`

Returns a read-only (constant) iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 333 of file stl\_multimap.h.

**5.604.3.11** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> iterator std::multimap< _Key, _Tp, _Compare,  
_Alloc >::end ( ) [inline]`

Returns a read/write iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 324 of file stl\_multimap.h.

**5.604.3.12** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> std::pair<iterator, iterator> std::multimap< _Key,  
_Tp, _Compare, _Alloc >::equal_range ( const key_type & __x )  
[inline]`

Finds a subsequence matching given key.

#### Parameters

*x* Key of (key, value) pairs to be located.

#### Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
 c.upper_bound(val))
```

(but is faster than making the calls separately).

Definition at line 752 of file stl\_multimap.h.

**5.604.3.13** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> std::pair<const_iterator, const_iterator>  
std::multimap<_Key, _Tp, _Compare, _Alloc>::equal_range (  
const key_type & __x ) const [inline]`

Finds a subsequence matching given key.

#### Parameters

*x* Key of (key, value) pairs to be located.

#### Returns

Pair of read-only (constant) iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
 c.upper_bound(val))
```

(but is faster than making the calls separately).

Definition at line 769 of file `stl_multimap.h`.

**5.604.3.14** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> iterator std::multimap<_Key, _Tp, _Compare,  
_Alloc>::erase ( const_iterator __position ) [inline]`

Erases an element from a multimap.

#### Parameters

*position* An iterator pointing to the element to be erased.

#### Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a multimap. Note that this function only erases the element, and that if the element is itself a pointer,

the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 534 of file `stl_multimap.h`.

**5.604.3.15** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> size_type std::multimap<_Key, _Tp, _Compare,  
_Alloc >::erase ( const key_type & __x ) [inline]`

Erases elements according to the provided key.

#### Parameters

*x* Key of element to be erased.

#### Returns

The number of elements erased.

This function erases all elements located by the given key from a multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 564 of file `stl_multimap.h`.

**5.604.3.16** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> iterator std::multimap<_Key, _Tp, _Compare,  
_Alloc >::erase ( const_iterator __first, const_iterator __last )  
[inline]`

Erases a [*first*,*last*) range of elements from a multimap.

#### Parameters

*first* Iterator pointing to the start of the range to be erased.

*last* Iterator pointing to the end of the range to be erased.

#### Returns

The iterator *last*.

This function erases a sequence of elements from a multimap. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 584 of file stl\_multimap.h.

**5.604.3.17** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> iterator std::multimap< _Key, _Tp, _Compare,  
_Alloc >::find ( const key_type & __x ) [inline]`

Tries to locate an element in a multimap.

#### Parameters

*x* Key of (key, value) pair to be located.

#### Returns

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 661 of file stl\_multimap.h.

**5.604.3.18** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> const_iterator std::multimap< _Key, _Tp,  
_Compare, _Alloc >::find ( const key_type & __x ) const  
[inline]`

Tries to locate an element in a multimap.

#### Parameters

*x* Key of (key, value) pair to be located.

#### Returns

Read-only (constant) iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns a constant iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 676 of file `stl_multimap.h`.

**5.604.3.19** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> allocator_type std::multimap< _Key, _Tp,  
_Compare, _Alloc >::get_allocator ( ) const [inline]`

Get a copy of the memory allocation object.

Definition at line 296 of file `stl_multimap.h`.

**5.604.3.20** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> iterator std::multimap< _Key, _Tp, _Compare,  
_Alloc >::insert ( const value_type & __x ) [inline]`

Inserts a [std::pair](#) into the multimap.

#### Parameters

*x* Pair to be inserted (see [std::make\\_pair](#) for easy creation of pairs).

#### Returns

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a [std::map](#) the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted.

Insertion requires logarithmic time.

Definition at line 440 of file `stl_multimap.h`.

Referenced by `std::multimap< _Key, _Tp, _Compare, _Alloc >::operator=()`.



**5.604.3.21** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> iterator std::multimap< _Key, _Tp, _Compare,  
_Alloc >::insert ( const_iterator __position, const value_type &  
__x ) [inline]`

Inserts a `std::pair` into the multimap.

#### Parameters

- position* An iterator that serves as a hint as to where the pair should be inserted.  
*x* Pair to be inserted (see `std::make_pair` for easy creation of pairs).

#### Returns

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a `std::map` the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.htm>

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 474 of file `stl_multimap.h`.

**5.604.3.22** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> template<typename _InputIterator > void  
std::multimap< _Key, _Tp, _Compare, _Alloc >::insert (   
_InputIterator __first, _InputIterator __last ) [inline]`

A template function that attempts to insert a range of elements.

#### Parameters

- first* Iterator pointing to the start of the range to be inserted.  
*last* Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 501 of file `stl_multimap.h`.

**5.604.3.23** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> void std::multimap< _Key, _Tp, _Compare, _Alloc  
>::insert ( initializer_list< value_type > __l ) [inline]`

Attempts to insert a list of std::pairs into the multimap.

#### Parameters

*list* A std::initializer\_list<value\_type> of pairs to be inserted.

Complexity similar to that of the range constructor.

Definition at line 513 of file stl\_multimap.h.

References std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >::insert().

Referenced by std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >::insert().

**5.604.3.24** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> key_compare std::multimap< _Key, _Tp,  
_Compare, _Alloc >::key_comp ( ) const [inline]`

Returns the key comparison object out of which the multimap was constructed.

Definition at line 637 of file stl\_multimap.h.

**5.604.3.25** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> iterator std::multimap< _Key, _Tp, _Compare,  
_Alloc >::lower_bound ( const key_type & __x ) [inline]`

Finds the beginning of a subsequence matching given key.

#### Parameters

*x* Key of (key, value) pair to be located.

#### Returns

Iterator pointing to first element equal to or greater than key, or [end\(\)](#).

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or [end\(\)](#) if no such element exists.

Definition at line 700 of file stl\_multimap.h.

**5.604.3.26** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> const_iterator std::multimap< _Key, _Tp,  
_Compare, _Alloc >::lower_bound ( const key_type & __x ) const  
[inline]`

Finds the beginning of a subsequence matching given key.

#### Parameters

*x* Key of (key, value) pair to be located.

#### Returns

Read-only (constant) iterator pointing to first element equal to or greater than key,  
or `end()`.

This function returns the first element of a subsequence of elements that matches the  
given key. If unsuccessful the iterator will point to the next greatest element or, if no  
such greater element exists, to `end()`.

Definition at line 715 of file stl\_multimap.h.

**5.604.3.27** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> size_type std::multimap< _Key, _Tp, _Compare,  
_Alloc >::max_size ( ) const [inline]`

Returns the maximum size of the multimap.

Definition at line 423 of file stl\_multimap.h.

**5.604.3.28** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> multimap& std::multimap< _Key, _Tp, _Compare,  
_Alloc >::operator= ( initializer_list< value_type > __l )  
[inline]`

Multimap list assignment operator.

### Parameters

*l* An [initializer\\_list](#).

This function fills a multimap with copies of the elements in the initializer list *l*.

Note that the assignment completely changes the multimap and that the resulting multimap's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 286 of file `std_multimap.h`.

References `std::multimap< _Key, _Tp, _Compare, _Alloc >::clear()`, and `std::multimap< _Key, _Tp, _Compare, _Alloc >::insert()`.

**5.604.3.29** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> multimap& std::multimap< _Key, _Tp, _Compare,  
_Alloc >::operator= ( multimap< _Key, _Tp, _Compare, _Alloc >  
&& __x ) [inline]`

Multimap move assignment operator.

### Parameters

*x* A multimap of identical element and allocator types.

The contents of *x* are moved into this multimap (without copying). *x* is a valid, but unspecified multimap.

Definition at line 265 of file `std_multimap.h`.

References `std::multimap< _Key, _Tp, _Compare, _Alloc >::clear()`, and `std::multimap< _Key, _Tp, _Compare, _Alloc >::swap()`.

**5.604.3.30** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> multimap& std::multimap< _Key, _Tp, _Compare,  
_Alloc >::operator= ( const multimap< _Key, _Tp, _Compare,  
_Alloc > & __x ) [inline]`

Multimap assignment operator.

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**Parameters**

*x* A multimap of identical element and allocator types.

All the elements of *x* are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 250 of file stl\_multimap.h.

**5.604.3.31** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> reverse_iterator std::multimap< _Key, _Tp,  
_Compare, _Alloc >::rbegin ( ) [inline]`

Returns a read/write reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 342 of file stl\_multimap.h.

**5.604.3.32** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> const_reverse_iterator std::multimap< _Key, _Tp,  
_Compare, _Alloc >::rbegin ( ) const [inline]`

Returns a read-only (constant) reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 351 of file stl\_multimap.h.

**5.604.3.33** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> reverse_iterator std::multimap< _Key, _Tp,  
_Compare, _Alloc >::rend ( ) [inline]`

Returns a read/write reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 360 of file stl\_multimap.h.

**5.604.3.34** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> const_reverse_iterator std::multimap< _Key, _Tp,  
_Compare, _Alloc >::rend ( ) const [inline]`

Returns a read-only (constant) reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 369 of file stl\_multimap.h.

**5.604.3.35** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> size_type std::multimap<_Key, _Tp, _Compare,  
_Alloc>::size ( ) const [inline]`

Returns the size of the multimap.

Definition at line 418 of file stl\_multimap.h.

**5.604.3.36** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> void std::multimap<_Key, _Tp, _Compare, _Alloc  
>::swap ( multimap<_Key, _Tp, _Compare, _Alloc> & __x )  
[inline]`

Swaps data with another multimap.

#### Parameters

*x* A multimap of the same element and allocator types.

This exchanges the elements between two multimaps in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(m1,m2)` will feed to this function.

Definition at line 618 of file stl\_multimap.h.

Referenced by `std::multimap< _Key, _Tp, _Compare, _Alloc >::operator=()`, and `std::swap()`.

**5.604.3.37** `template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp>>> iterator std::multimap<_Key, _Tp, _Compare,  
_Alloc>::upper_bound ( const key_type & __x ) [inline]`

Finds the end of a subsequence matching given key.

#### Parameters

*x* Key of (key, value) pair to be located.

**Returns**

Iterator pointing to the first element greater than key, or [end\(\)](#).

Definition at line 725 of file `stl_multimap.h`.

```
5.604.3.38 template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> const_iterator std::multimap<_Key, _Tp,
_Compare, _Alloc>::upper_bound (const key_type & __x) const
[inline]
```

Finds the end of a subsequence matching given key.

**Parameters**

*x* Key of (key, value) pair to be located.

**Returns**

Read-only (constant) iterator pointing to first iterator greater than key, or [end\(\)](#).

Definition at line 735 of file `stl_multimap.h`.

```
5.604.3.39 template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> value_compare std::multimap<_Key, _Tp,
_Compare, _Alloc>::value_comp () const [inline]
```

Returns a value comparison object, built from the key comparison object out of which the multimap was constructed.

Definition at line 645 of file `stl_multimap.h`.

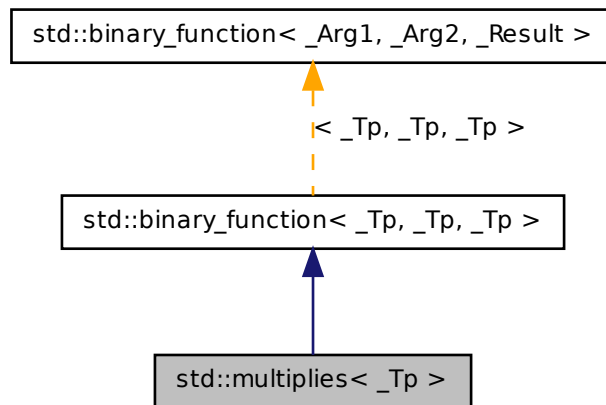
The documentation for this class was generated from the following file:

- [stl\\_multimap.h](#)

**5.605 `std::multiplies<_Tp>` Struct Template Reference**

One of the [math functors](#).

Inheritance diagram for `std::multiplies<_Tp>`:



### Public Types

- typedef `_Tp` `first_argument_type`
- typedef `_Tp` `result_type`
- typedef `_Tp` `second_argument_type`

### Public Member Functions

- `_Tp operator() (const _Tp &__x, const _Tp &__y) const`

#### 5.605.1 Detailed Description

**template<typename \_Tp> struct `std::multiplies<_Tp>`**

One of the [math functors](#).

Definition at line 159 of file `stl_function.h`.



## 5.606 `std::multiset< _Key, _Compare, _Alloc >` Class Template Reference 3006

---

### 5.605.2 Member Typedef Documentation

**5.605.2.1** `typedef _Tp std::binary_function< _Tp, _Tp, _Tp  
>::first_argument_type [inherited]`

`first_argument_type` is the type of the first argument

Definition at line 118 of file `stl_function.h`.

**5.605.2.2** `typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::result_type  
[inherited]`

`result_type` is the return type

Definition at line 124 of file `stl_function.h`.

**5.605.2.3** `typedef _Tp std::binary_function< _Tp, _Tp, _Tp  
>::second_argument_type [inherited]`

`second_argument_type` is the type of the second argument

Definition at line 121 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.606 `std::multiset< _Key, _Compare, _Alloc >` Class Template Reference

A standard container made up of elements, which can be retrieved in logarithmic time.

### Public Types

- `typedef _Alloc allocator_type`
- `typedef _Rep_type::const_iterator const_iterator`
- `typedef _Key_alloc_type::const_pointer const_pointer`
- `typedef _Key_alloc_type::const_reference const_reference`
- `typedef _Rep_type::const_reverse_iterator const_reverse_iterator`
- `typedef _Rep_type::difference_type difference_type`

- typedef `_Rep_type::const_iterator` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Key_alloc_type::pointer` **pointer**
- typedef `_Key_alloc_type::reference` **reference**
- typedef `_Rep_type::const_reverse_iterator` **reverse\_iterator**
- typedef `_Rep_type::size_type` **size\_type**
- typedef `_Compare` **value\_compare**
- typedef `_Key` **value\_type**

### Public Member Functions

- [multiset](#) ()
- [multiset](#) (const `_Compare` &\_\_comp, const `allocator_type` &\_\_a=allocator\_type())
- `template<typename _InputIterator >`  
[multiset](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, const `_Compare` &\_\_comp, const `allocator_type` &\_\_a=allocator\_type())
- [multiset](#) (const [multiset](#) &\_\_x)
- `template<typename _InputIterator >`  
[multiset](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- [multiset](#) ([multiset](#) &&\_\_x)
- [multiset](#) ([initializer\\_list](#)< `value_type` > \_\_l, const `_Compare` &\_\_comp=\_Compare(), const `allocator_type` &\_\_a=allocator\_type())
- iterator [begin](#) () const
- iterator [cbegin](#) () const
- iterator [cend](#) () const
- void [clear](#) ()
- `size_type` [count](#) (const `key_type` &\_\_x) const
- `reverse_iterator` [crbegin](#) () const
- `reverse_iterator` [crend](#) () const
- bool [empty](#) () const
- iterator [end](#) () const
- `size_type` [erase](#) (const `key_type` &\_\_x)
- iterator [erase](#) (const\_iterator \_\_position)
- iterator [erase](#) (const\_iterator \_\_first, const\_iterator \_\_last)
- `allocator_type` [get\\_allocator](#) () const
- iterator **insert** (const\_iterator \_\_position, `value_type` &&\_\_x)
- iterator **insert** (`value_type` &&\_\_x)
- void [insert](#) ([initializer\\_list](#)< `value_type` > \_\_l)
- iterator [insert](#) (const\_iterator \_\_position, const `value_type` &\_\_x)
- iterator [insert](#) (const `value_type` &\_\_x)

- `template<typename _InputIterator >`  
`void insert (_InputIterator __first, _InputIterator __last)`
- `key_compare key_comp () const`
- `size_type max_size () const`
- `multiset & operator= (const multiset &__x)`
- `multiset & operator= (multiset &&__x)`
- `multiset & operator= (initializer_list< value_type > __l)`
- `reverse_iterator rbegin () const`
- `reverse_iterator rend () const`
- `size_type size () const`
- `void swap (multiset &__x)`
- `value_compare value_comp () const`
  
- `iterator find (const key_type &__x)`
- `const_iterator find (const key_type &__x) const`
  
- `iterator lower_bound (const key_type &__x)`
- `const_iterator lower_bound (const key_type &__x) const`
  
- `iterator upper_bound (const key_type &__x)`
- `const_iterator upper_bound (const key_type &__x) const`
  
- `std::pair< iterator, iterator > equal_range (const key_type &__x)`
- `std::pair< const_iterator, const_iterator > equal_range (const key_type &__x)`  
`const`
- `template<typename _K1, typename _C1, typename _A1 >`  
`bool operator== (const multiset< _K1, _C1, _A1 > &, const multiset< _K1, _C1, _A1 > &)`
- `template<typename _K1, typename _C1, typename _A1 >`  
`bool operator< (const multiset< _K1, _C1, _A1 > &, const multiset< _K1, _C1, _A1 > &)`

### 5.606.1 Detailed Description

`template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> class std::multiset<_Key, _Compare, _Alloc>`

A standard container made up of elements, which can be retrieved in logarithmic time. Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using equivalent keys). For a `multiset<Key>` the `key_type` and `value_type` are `Key`.

Multisets support bidirectional iterators.

## 5.606 std::multiset<\_Key, \_Compare, \_Alloc > Class Template Reference 3009

---

The private tree data is declared exactly the same way for set and multiset; the distinction is made entirely in how the tree functions are called (\*\_unique versus \*\_equal, same as the standard).

Definition at line 86 of file stl\_multiset.h.

### 5.606.2 Constructor & Destructor Documentation

**5.606.2.1** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> std::multiset<_Key,  
_Compare, _Alloc>::multiset ( ) [inline]`

Default constructor creates no elements.

Definition at line 131 of file stl\_multiset.h.

**5.606.2.2** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> std::multiset<_Key,  
_Compare, _Alloc>::multiset ( const _Compare & __comp, const  
allocator_type & __a = allocator_type() ) [inline,  
explicit]`

Creates a multiset with no elements.

#### Parameters

*comp* Comparator to use.

*a* An allocator object.

Definition at line 140 of file stl\_multiset.h.

**5.606.2.3** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> template<typename  
_InputIterator > std::multiset<_Key, _Compare, _Alloc>::multiset  
( _InputIterator __first, _InputIterator __last ) [inline]`

Builds a multiset from a range.

#### Parameters

*first* An input iterator.

## 5.606 std::multiset<\_Key, \_Compare, \_Alloc> Class Template Reference 3010

---

*last* An input iterator.

Create a multiset consisting of copies of the elements from [first,last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(first,last)).

Definition at line 154 of file stl\_multiset.h.

```
5.606.2.4 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> template<typename
_InputIterator > std::multiset<_Key, _Compare, _Alloc>::multiset
(_InputIterator __first, _InputIterator __last, const _Compare &
__comp, const allocator_type & __a = allocator_type())
[inline]
```

Builds a multiset from a range.

### Parameters

*first* An input iterator.

*last* An input iterator.

*comp* A comparison functor.

*a* An allocator object.

Create a multiset consisting of copies of the elements from [first,last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(first,last)).

Definition at line 170 of file stl\_multiset.h.

```
5.606.2.5 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> std::multiset<_Key,
_Compare, _Alloc>::multiset (const multiset<_Key, _Compare,
_Alloc> & __x) [inline]
```

Multiset copy constructor.

### Parameters

*x* A multiset of identical element and allocator types.

The newly-created multiset uses a copy of the allocation object used by *x*.

Definition at line 183 of file stl\_multiset.h.

**5.606.2.6** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> std::multiset< _Key,  
_Compare, _Alloc >::multiset ( multiset< _Key, _Compare, _Alloc >  
&& __x ) [inline]`

Multiset move constructor.

#### Parameters

*x* A multiset of identical element and allocator types.

The newly-created multiset contains the exact contents of *x*. The contents of *x* are a valid, but unspecified multiset.

Definition at line 194 of file stl\_multiset.h.

**5.606.2.7** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> std::multiset< _Key,  
_Compare, _Alloc >::multiset ( initializer_list< value_type > __l,  
const _Compare & __comp = _Compare(), const allocator_type &  
__a = allocator_type() ) [inline]`

Builds a multiset from an [initializer\\_list](#).

#### Parameters

*l* An [initializer\\_list](#).

*comp* A comparison functor.

*a* An allocator object.

Create a multiset consisting of copies of the elements from the list. This is linear in *N* if the list is already sorted, and *NlogN* otherwise (where *N* is *l.size()*).

Definition at line 207 of file stl\_multiset.h.

### 5.606.3 Member Function Documentation

**5.606.3.1** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> iterator std::multiset<  
_Key, _Compare, _Alloc >::begin ( ) const [inline]`

Returns a read-only (constant) iterator that points to the first element in the multiset. Iteration is done in ascending order according to the keys.

## 5.606 std::multiset< \_Key, \_Compare, \_Alloc > Class Template Reference 3012

Definition at line 287 of file stl\_multiset.h.

**5.606.3.2** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> iterator std::multiset<  
_Key, _Compare, _Alloc >::cbegin ( ) const [inline]`

Returns a read-only (constant) iterator that points to the first element in the multiset. Iteration is done in ascending order according to the keys.

Definition at line 324 of file stl\_multiset.h.

**5.606.3.3** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> iterator std::multiset<  
_Key, _Compare, _Alloc >::cend ( ) const [inline]`

Returns a read-only (constant) iterator that points one past the last element in the multiset. Iteration is done in ascending order according to the keys.

Definition at line 333 of file stl\_multiset.h.

**5.606.3.4** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> void std::multiset< _Key,  
_Compare, _Alloc >::clear ( ) [inline]`

Erases all elements in a multiset. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 557 of file stl\_multiset.h.

Referenced by std::multiset< \_Key, \_Compare, \_Alloc >::operator=().

**5.606.3.5** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> size_type std::multiset<  
_Key, _Compare, _Alloc >::count ( const key_type & __x ) const  
[inline]`

Finds the number of elements with given key.

### Parameters

*x* Key of elements to be located.

## 5.606 std::multiset<\_Key, \_Compare, \_Alloc > Class Template Reference 3013

---

### Returns

Number of elements with specified key.

Definition at line 568 of file stl\_multiset.h.

**5.606.3.6** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> reverse_iterator  
std::multiset<_Key, _Compare, _Alloc >::crbegin ( ) const  
[inline]`

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 342 of file stl\_multiset.h.

**5.606.3.7** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> reverse_iterator  
std::multiset<_Key, _Compare, _Alloc >::crend ( ) const  
[inline]`

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 351 of file stl\_multiset.h.

**5.606.3.8** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> bool std::multiset<_Key,  
_Compare, _Alloc >::empty ( ) const [inline]`

Returns true if the set is empty.

Definition at line 357 of file stl\_multiset.h.

**5.606.3.9** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> iterator std::multiset<  
_Key, _Compare, _Alloc >::end ( ) const [inline]`

Returns a read-only (constant) iterator that points one past the last element in the multiset. Iteration is done in ascending order according to the keys.

Definition at line 296 of file stl\_multiset.h.



**5.606.3.10** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> std::pair<iterator,  
iterator> std::multiset<_Key, _Compare, _Alloc>::equal_range (  
const key_type & __x ) [inline]`

Finds a subsequence matching given key.

**Parameters**

*x* Key to be located.

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
 c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 648 of file `std_multiset.h`.

**5.606.3.11** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> std::pair<const_iterator,  
const_iterator> std::multiset<_Key, _Compare, _Alloc  
>::equal_range ( const key_type & __x ) const [inline]`

Finds a subsequence matching given key.

**Parameters**

*x* Key to be located.

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
 c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 652 of file stl\_multiset.h.

**5.606.3.12** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> iterator std::multiset<  
_Key, _Compare, _Alloc >::erase ( const_iterator __position )  
[inline]`

Erases an element from a multiset.

#### Parameters

*position* An iterator pointing to the element to be erased.

#### Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, [end\(\)](#) is returned.

This function erases an element, pointed to by the given iterator, from a multiset. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 480 of file stl\_multiset.h.

**5.606.3.13** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> size_type std::multiset<  
_Key, _Compare, _Alloc >::erase ( const key_type & __x )  
[inline]`

Erases elements according to the provided key.

#### Parameters

*x* Key of element to be erased.

#### Returns

The number of elements erased.

## 5.606 std::multiset< \_Key, \_Compare, \_Alloc > Class Template Reference 3016

This function erases all elements located by the given key from a multiset. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 510 of file stl\_multiset.h.

```
5.606.3.14 template<typename _Key, typename _Compare = std::less<_Key>,
 typename _Alloc = std::allocator<_Key>> iterator std::multiset<
 _Key, _Compare, _Alloc >::erase (const_iterator __first,
 const_iterator __last) [inline]
```

Erases a [first,last) range of elements from a multiset.

### Parameters

*first* Iterator pointing to the start of the range to be erased.

*last* Iterator pointing to the end of the range to be erased.

### Returns

The iterator *last*.

This function erases a sequence of elements from a multiset. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 530 of file stl\_multiset.h.

```
5.606.3.15 template<typename _Key, typename _Compare = std::less<_Key>,
 typename _Alloc = std::allocator<_Key>> iterator std::multiset<
 _Key, _Compare, _Alloc >::find (const key_type & __x)
 [inline]
```

Tries to locate an element in a set.

### Parameters

*x* Element to be located.

### Returns

Iterator pointing to sought-after element, or [end\(\)](#) if not found.

## 5.606 std::multiset< \_Key, \_Compare, \_Alloc > Class Template Reference 3017

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 586 of file `stl_multiset.h`.

**5.606.3.16** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> const_iterator  
std::multiset< _Key, _Compare, _Alloc >::find ( const key_type &  
__x ) const [inline]`

Tries to locate an element in a set.

### Parameters

`x` Element to be located.

### Returns

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 590 of file `stl_multiset.h`.

**5.606.3.17** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> allocator_type  
std::multiset< _Key, _Compare, _Alloc >::get_allocator ( ) const  
[inline]`

Returns the memory allocation object.

Definition at line 278 of file `stl_multiset.h`.

**5.606.3.18** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> iterator std::multiset<  
_Key, _Compare, _Alloc >::insert ( const value_type & __x )  
[inline]`

Inserts an element into the multiset.

## 5.606 `std::multiset<_Key, _Compare, _Alloc>` Class Template Reference 3018

---

### Parameters

*x* Element to be inserted.

### Returns

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a [std::set](#) the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.

Insertion requires logarithmic time.

Definition at line 398 of file `stl_multiset.h`.

Referenced by `std::multiset<_Key, _Compare, _Alloc>::operator=()`.

```
5.606.3.19 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::multiset<
_Key, _Compare, _Alloc>::insert (const_iterator __position, const
value_type & __x) [inline]
```

Inserts an element into the multiset.

### Parameters

*position* An iterator that serves as a hint as to where the element should be inserted.

*x* Element to be inserted.

### Returns

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a [std::set](#) the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html> for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 428 of file `stl_multiset.h`.

**5.606.3.20** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> template<typename  
_InputIterator > void std::multiset< _Key, _Compare, _Alloc  
>::insert ( _InputIterator __first, _InputIterator __last )  
[inline]`

A template function that tries to insert a range of elements.

#### Parameters

*first* Iterator pointing to the start of the range to be inserted.

*last* Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 447 of file stl\_multiset.h.

**5.606.3.21** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> void std::multiset<  
_Key, _Compare, _Alloc >::insert ( initializer_list< value_type >  
__l ) [inline]`

Attempts to insert a list of elements into the multiset.

#### Parameters

*list* A std::initializer\_list<value\_type> of elements to be inserted.

Complexity similar to that of the range constructor.

Definition at line 459 of file stl\_multiset.h.

References std::multiset< \_Key, \_Compare, \_Alloc >::insert().

Referenced by std::multiset< \_Key, \_Compare, \_Alloc >::insert().

**5.606.3.22** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> key_compare  
std::multiset< _Key, _Compare, _Alloc >::key_comp ( ) const  
[inline]`

Returns the comparison object.

Definition at line 270 of file stl\_multiset.h.

**5.606.3.23** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> iterator std::multiset<  
_Key, _Compare, _Alloc >::lower_bound ( const key_type & __x )  
[inline]`

Finds the beginning of a subsequence matching given key.

#### Parameters

*x* Key to be located.

#### Returns

Iterator pointing to first element equal to or greater than key, or `end()`.

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or `end()` if no such element exists.

Definition at line 607 of file `stl_multiset.h`.

**5.606.3.24** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> const_iterator  
std::multiset< _Key, _Compare, _Alloc >::lower_bound ( const  
key_type & __x ) const [inline]`

Finds the beginning of a subsequence matching given key.

#### Parameters

*x* Key to be located.

#### Returns

Iterator pointing to first element equal to or greater than key, or `end()`.

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or `end()` if no such element exists.

Definition at line 611 of file `stl_multiset.h`.

## 5.606 std::multiset< \_Key, \_Compare, \_Alloc > Class Template Reference 3021

**5.606.3.25** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> size_type std::multiset<  
_Key, _Compare, _Alloc >::max_size ( ) const [inline]`

Returns the maximum size of the set.

Definition at line 367 of file stl\_multiset.h.

**5.606.3.26** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> multiset&  
std::multiset< _Key, _Compare, _Alloc >::operator= ( multiset<  
_Key, _Compare, _Alloc > && __x ) [inline]`

Multiset move assignment operator.

### Parameters

*x* A multiset of identical element and allocator types.

The contents of *x* are moved into this multiset (without copying). *x* is a valid, but unspecified multiset.

Definition at line 237 of file stl\_multiset.h.

References std::multiset< \_Key, \_Compare, \_Alloc >::clear(), and std::multiset< \_Key, \_Compare, \_Alloc >::swap().

**5.606.3.27** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> multiset&  
std::multiset< _Key, _Compare, _Alloc >::operator= ( const  
multiset< _Key, _Compare, _Alloc > & __x ) [inline]`

Multiset assignment operator.

### Parameters

*x* A multiset of identical element and allocator types.

All the elements of *x* are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 222 of file stl\_multiset.h.



**5.606.3.28** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> multiset&  
std::multiset< _Key, _Compare, _Alloc >::operator= (  
initializer_list< value_type > __l ) [inline]`

Multiset list assignment operator.

#### Parameters

*l* An [initializer\\_list](#).

This function fills a multiset with copies of the elements in the initializer list *l*.

Note that the assignment completely changes the multiset and that the resulting multiset's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 258 of file `stl_multiset.h`.

References `std::multiset< _Key, _Compare, _Alloc >::clear()`, and `std::multiset< _Key, _Compare, _Alloc >::insert()`.

**5.606.3.29** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> reverse_iterator  
std::multiset< _Key, _Compare, _Alloc >::rbegin ( ) const  
[inline]`

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 305 of file `stl_multiset.h`.

**5.606.3.30** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> reverse_iterator  
std::multiset< _Key, _Compare, _Alloc >::rend ( ) const  
[inline]`

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 314 of file `stl_multiset.h`.

**5.606.3.31** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> size_type std::multiset<  
_Key, _Compare, _Alloc >::size ( ) const [inline]`

Returns the size of the set.

Definition at line 362 of file stl\_multiset.h.

**5.606.3.32** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> void std::multiset<  
_Key, _Compare, _Alloc >::swap ( multiset< _Key, _Compare,  
_Alloc > & __x ) [inline]`

Swaps data with another multiset.

#### Parameters

*x* A multiset of the same element and allocator types.

This exchanges the elements between two multisets in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(s1,s2)` will feed to this function.

Definition at line 382 of file stl\_multiset.h.

Referenced by `std::multiset< _Key, _Compare, _Alloc >::operator=()`, and `std::swap()`.

**5.606.3.33** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> const_iterator  
std::multiset< _Key, _Compare, _Alloc >::upper_bound ( const  
key_type & __x ) const [inline]`

Finds the end of a subsequence matching given key.

#### Parameters

*x* Key to be located.

#### Returns

Iterator pointing to the first element greater than key, or `end()`.

Definition at line 627 of file stl\_multiset.h.

**5.606.3.34** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> iterator std::multiset<  
_Key, _Compare, _Alloc >::upper_bound ( const key_type & __x )  
[inline]`

Finds the end of a subsequence matching given key.

**Parameters**

*x* Key to be located.

**Returns**

Iterator pointing to the first element greater than key, or [end\(\)](#).

Definition at line 623 of file stl\_multiset.h.

**5.606.3.35** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> value_compare  
std::multiset< _Key, _Compare, _Alloc >::value_comp ( ) const  
[inline]`

Returns the comparison object.

Definition at line 274 of file stl\_multiset.h.

**5.606.4 Friends And Related Function Documentation**

**5.606.4.1** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> template<typename _K1  
, typename _C1, typename _A1 > bool operator< ( const multiset<  
_K1, _C1, _A1 > &, const multiset< _K1, _C1, _A1 > & )  
[friend]`

Finds a subsequence matching given key.

**Parameters**

*x* Key to be located.

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
 c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

**5.606.4.2** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> template<typename  
_K1, typename _C1, typename _A1 > bool operator==( const  
multiset<_K1, _C1, _A1 > &, const multiset<_K1, _C1, _A1 > &  
) [friend]`

Finds a subsequence matching given key.

#### Parameters

*x* Key to be located.

#### Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
 c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

The documentation for this class was generated from the following file:

- [stl\\_multiset.h](#)

## 5.607 std::mutex Class Reference

mutex

#### Public Types

- typedef \_\_native\_type \* native\_handle\_type

**Public Member Functions**

- `mutex` (const `mutex` &)
- `void lock` ()
- `native_handle_type native_handle` ()
- `mutex` & `operator=` (const `mutex` &)
- `bool try_lock` ()
- `void unlock` ()

**5.607.1 Detailed Description**

`mutex`

Definition at line 64 of file `mutex`.

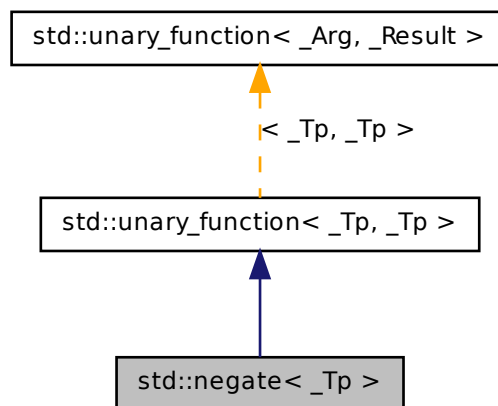
The documentation for this class was generated from the following file:

- `mutex`

**5.608 `std::negate<_Tp>` Struct Template Reference**

One of the [math functors](#).

Inheritance diagram for `std::negate<_Tp>`:



## Public Types

- typedef `_Tp` [argument\\_type](#)
- typedef `_Tp` [result\\_type](#)

## Public Member Functions

- `_Tp` **operator()** (const `_Tp` &\_\_x) const

### 5.608.1 Detailed Description

`template<typename _Tp> struct std::negate<_Tp>`

One of the [math functors](#).

Definition at line 186 of file `stl_function.h`.

### 5.608.2 Member Typedef Documentation

#### 5.608.2.1 `typedef _Tp std::unary_function<_Tp, _Tp>::argument_type` [[inherited](#)]

`argument_type` is the type of the argument

Definition at line 105 of file `stl_function.h`.

#### 5.608.2.2 `typedef _Tp std::unary_function<_Tp, _Tp>::result_type` [[inherited](#)]

`result_type` is the return type

Definition at line 108 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.609 `std::negative_binomial_distribution<_IntType>` Class Template Reference

A [negative\\_binomial\\_distribution](#) random number distribution.

## Classes

- struct `param_type`

## Public Types

- typedef `_IntType` `result_type`

## Public Member Functions

- **`negative_binomial_distribution`** (`_IntType` `__k`=1, double `__p`=0.5)
- **`negative_binomial_distribution`** (const `param_type` &`__p`)
- `_IntType` `k` () const
- `result_type` `max` () const
- `result_type` `min` () const
- template<typename `_UniformRandomNumberGenerator` >  
`result_type` **`operator()`** (`_UniformRandomNumberGenerator` &`__urng`, const `param_type` &`__p`)
- template<typename `_UniformRandomNumberGenerator` >  
`result_type` **`operator()`** (`_UniformRandomNumberGenerator` &`__urng`)
- double `p` () const
- void `param` (const `param_type` &`__param`)
- `param_type` `param` () const
- void `reset` ()

## Friends

- template<typename `_IntType1`, typename `_CharT`, typename `_Traits` >  
`std::basic_ostream`< `_CharT`, `_Traits` > & **`operator<<`** (`std::basic_ostream`< `_CharT`, `_Traits` > &, const `std::negative_binomial_distribution`< `_IntType1` > &)
- template<typename `_IntType1` >  
bool **`operator==`** (const `std::negative_binomial_distribution`< `_IntType1` > &`__d1`, const `std::negative_binomial_distribution`< `_IntType1` > &`__d2`)
- template<typename `_IntType1`, typename `_CharT`, typename `_Traits` >  
`std::basic_istream`< `_CharT`, `_Traits` > & **`operator>>`** (`std::basic_istream`< `_CharT`, `_Traits` > &, `std::negative_binomial_distribution`< `_IntType1` > &)

### 5.609.1 Detailed Description

`template<typename _IntType = int> class std::negative_binomial_distribution<_IntType>`

A [negative\\_binomial\\_distribution](#) random number distribution. The formula for the negative binomial probability mass function is  $p(i) = \binom{n}{i} p^i (1-p)^{t-i}$  where  $t$  and  $p$  are the parameters of the distribution.

Definition at line 3769 of file random.h.

### 5.609.2 Member Typedef Documentation

5.609.2.1 `template<typename _IntType = int> typedef _IntType  
std::negative_binomial_distribution<_IntType>::result_type`

The type of the range of the distribution.

Definition at line 3776 of file random.h.

### 5.609.3 Member Function Documentation

5.609.3.1 `template<typename _IntType = int> _IntType  
std::negative_binomial_distribution<_IntType>::k ( ) const  
[inline]`

Return the  $k$  parameter of the distribution.

Definition at line 3825 of file random.h.

5.609.3.2 `template<typename _IntType = int> result_type  
std::negative_binomial_distribution<_IntType>::max ( ) const  
[inline]`

Returns the least upper bound value of the distribution.

Definition at line 3861 of file random.h.



**5.609.3.3** `template<typename _IntType = int> result_type  
std::negative_binomial_distribution<_IntType>::min ( ) const  
[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 3854 of file random.h.

**5.609.3.4** `template<typename _IntType> template<typename  
_UniformRandomNumberGenerator> negative_  
binomial_distribution<_IntType>::result_type  
std::negative_binomial_distribution<_IntType>::operator() (  
_UniformRandomNumberGenerator & __urng )`

Generating functions.

Definition at line 1083 of file random.tcc.

**5.609.3.5** `template<typename _IntType = int> double  
std::negative_binomial_distribution<_IntType>::p ( ) const  
[inline]`

Return the *p* parameter of the distribution.

Definition at line 3832 of file random.h.

**5.609.3.6** `template<typename _IntType = int> param_type  
std::negative_binomial_distribution<_IntType>::param ( ) const  
[inline]`

Returns the parameter set of the distribution.

Definition at line 3839 of file random.h.

**5.609.3.7** `template<typename _IntType = int> void std::negative_binomial_  
distribution<_IntType>::param ( const param_type & __param )  
[inline]`

Sets the parameter set of the distribution.

**Parameters**

***\_\_param*** The new parameter set of the distribution.

Definition at line 3847 of file random.h.

**5.609.3.8** `template<typename _IntType = int> void  
std::negative_binomial_distribution< _IntType >::reset ( )  
[inline]`

Resets the distribution state.

Definition at line 3818 of file random.h.

References `std::gamma_distribution< _RealType >::reset()`.

**5.609.4 Friends And Related Function Documentation**

**5.609.4.1** `template<typename _IntType = int> template<typename _IntType1 ,  
typename _CharT , typename _Traits > std::basic_ostream<_CharT,  
_Traits>& operator<< ( std::basic_ostream< _CharT, _Traits > &  
, const std::negative_binomial_distribution< _IntType1 > & )  
[friend]`

Inserts a `negative_binomial_distribution` random number distribution `__x` into the output stream `__os`.

**Parameters**

***\_\_os*** An output stream.

***\_\_x*** A `negative_binomial_distribution` random number distribution.

**Returns**

The output stream with the state of `__x` inserted or in an error state.

5.609.4.2 `template<typename _IntType = int> template<typename _IntType1  
> bool operator==( const std::negative_binomial_distribution<  
_IntType1 > & __d1, const std::negative_binomial_distribution<  
_IntType1 > & __d2 ) [friend]`

Return true if two negative binomial distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 3883 of file `random.h`.

5.609.4.3 `template<typename _IntType = int> template<typename _IntType1 ,  
typename _CharT , typename _Traits > std::basic_istream<_CharT,  
_Traits>& operator>> ( std::basic_istream<_CharT, _Traits > & ,  
std::negative_binomial_distribution<_IntType1 > & ) [friend]`

Extracts a `negative_binomial_distribution` random number distribution `__x` from the input stream `__is`.

#### Parameters

- `__is` An input stream.
- `__x` A `negative_binomial_distribution` random number generator engine.

#### Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.610 `std::negative_binomial_distribution<_IntType>::param_type` Struct Reference

#### Public Types

- typedef [negative\\_binomial\\_distribution<\\_IntType>](#) `distribution_type`

#### Public Member Functions

- `param_type` (`_IntType __k=1`, `double __p=0.5`)

- `_IntType k () const`
- `double p () const`

### Friends

- `bool operator== (const param\_type &__p1, const param\_type &__p2)`

#### 5.610.1 Detailed Description

`template<typename _IntType = int> struct std::negative_binomial_distribution< _IntType >::param_type`

Parameter type.

Definition at line 3778 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.611 `std::nested_exception` Class Reference

Exception class with `exception_ptr` data member.

Inherited by `std::_Nested_exception< _Except >`.

### Public Member Functions

- `nested_exception (const nested\_exception &)`
- `exception_ptr nested_ptr () const`
- `nested\_exception & operator= (const nested\_exception &)`
- `void rethrow_nested () const __attribute__((__noreturn__))`

#### 5.611.1 Detailed Description

Exception class with `exception_ptr` data member.

Definition at line 55 of file `nested_exception.h`.

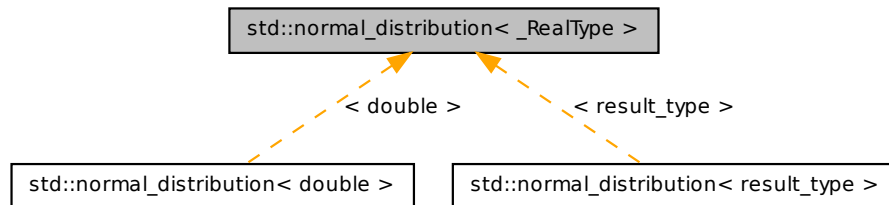
The documentation for this class was generated from the following file:

- [nested\\_exception.h](#)

## 5.612 `std::normal_distribution< _RealType >` Class Template Reference

A normal continuous distribution for random numbers.

Inheritance diagram for `std::normal_distribution< _RealType >`:



### Classes

- struct `param_type`

### Public Types

- typedef `_RealType` `result_type`

### Public Member Functions

- `normal_distribution` (`result_type` `__mean=result_type(0)`, `result_type` `__stddev=result_type(1)`)
- `normal_distribution` (const `param_type` &`__p`)
- `result_type max` () const
- `_RealType mean` () const
- `result_type min` () const
- template<typename `_UniformRandomNumberGenerator` >  
`result_type operator()` (`_UniformRandomNumberGenerator` &`__urng`, const `param_type` &`__p`)
- template<typename `_UniformRandomNumberGenerator` >  
`result_type operator()` (`_UniformRandomNumberGenerator` &`__urng`)
- void `param` (const `param_type` &`__param`)
- `param_type param` () const

- void `reset()`
- `_RealType stddev()` const

### Friends

- `template<typename _RealType1, typename _CharT, typename _Traits>`  
`std::basic_ostream<_CharT, _Traits> & operator<< (std::basic_ostream<_CharT, _Traits> &, const std::normal_distribution<_RealType1> &)`
- `template<typename _RealType1>`  
`bool operator==(const std::normal_distribution<_RealType1> &__d1, const std::normal_distribution<_RealType1> &__d2)`
- `template<typename _RealType1, typename _CharT, typename _Traits>`  
`std::basic_istream<_CharT, _Traits> & operator>> (std::basic_istream<_CharT, _Traits> &, std::normal_distribution<_RealType1> &)`

#### 5.612.1 Detailed Description

`template<typename _RealType = double> class std::normal_distribution<_RealType>`

A normal continuous distribution for random numbers. The formula for the normal probability density function is

$$p(x|\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x-\mu}{2\sigma^2}}$$

Definition at line 1967 of file random.h.

#### 5.612.2 Member Typedef Documentation

**5.612.2.1** `template<typename _RealType = double> typedef _RealType std::normal_distribution<_RealType>::result_type`

The type of the range of the distribution.

Definition at line 1974 of file random.h.

### 5.612.3 Constructor & Destructor Documentation

**5.612.3.1** `template<typename _RealType = double> std::normal_distribution<_RealType>::normal_distribution ( result_type __mean = result_type (0), result_type __stddev = result_type (1) )  
[inline, explicit]`

Constructs a normal distribution with parameters *mean* and standard deviation.  
Definition at line 2012 of file random.h.

### 5.612.4 Member Function Documentation

**5.612.4.1** `template<typename _RealType = double> result_type  
std::normal_distribution<_RealType>::max ( ) const [inline]`

Returns the least upper bound value of the distribution.  
Definition at line 2069 of file random.h.  
Referenced by std::normal\_distribution< result\_type >::max().

**5.612.4.2** `template<typename _RealType = double> _RealType  
std::normal_distribution<_RealType>::mean ( ) const  
[inline]`

Returns the mean of the distribution.  
Definition at line 2033 of file random.h.

**5.612.4.3** `template<typename _RealType = double> result_type  
std::normal_distribution<_RealType>::min ( ) const [inline]`

Returns the greatest lower bound value of the distribution.  
Definition at line 2062 of file random.h.  
Referenced by std::normal\_distribution< result\_type >::min().

**5.612.4.4** `template<typename _RealType = double> template<typename  
_UniformRandomNumberGenerator > result_type  
std::normal_distribution<_RealType>::operator() (  
_UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 2077 of file `random.h`.

Referenced by `std::normal_distribution<result_type>::operator()()`.

**5.612.4.5** `template<typename _RealType > template<typename  
_UniformRandomNumberGenerator > normal_distribution<  
_RealType>::result_type std::normal_distribution<_RealType  
>::operator() ( _UniformRandomNumberGenerator & __urng,  
const param_type & __param )`

Polar method due to Marsaglia.

Devroye, L. Non-Uniform Random Variates Generation. Springer-Verlag, New York, 1986, Ch. V, Sect. 4.4.

Definition at line 1649 of file `random.tcc`.

References `std::log()`, and `std::sqrt()`.

**5.612.4.6** `template<typename _RealType = double> void  
std::normal_distribution<_RealType>::param ( const param_type  
& __param ) [inline]`

Sets the parameter set of the distribution.

#### Parameters

`__param` The new parameter set of the distribution.

Definition at line 2055 of file `random.h`.

**5.612.4.7** `template<typename _RealType = double> param_type  
std::normal_distribution<_RealType>::param ( ) const  
[inline]`



Returns the parameter set of the distribution.

Definition at line 2047 of file random.h.

Referenced by std::normal\_distribution< result\_type >::operator()().

**5.612.4.8    template<typename \_RealType = double> void  
              std::normal\_distribution< \_RealType >::reset ( ) [inline]**

Resets the distribution state.

Definition at line 2026 of file random.h.

Referenced by std::poisson\_distribution< \_IntType >::reset(), std::binomial\_distribution< \_IntType >::reset(), std::student\_t\_distribution< \_RealType >::reset(), std::gamma\_distribution< result\_type >::reset(), and std::lognormal\_distribution< \_RealType >::reset().

**5.612.4.9    template<typename \_RealType = double> \_RealType  
              std::normal\_distribution< \_RealType >::stddev ( ) const  
              [inline]**

Returns the standard deviation of the distribution.

Definition at line 2040 of file random.h.

## **5.612.5 Friends And Related Function Documentation**

**5.612.5.1    template<typename \_RealType = double> template<typename  
              \_RealType1 , typename \_CharT , typename \_Traits >  
              std::basic\_ostream< \_CharT, \_Traits>& operator<<  
              ( std::basic\_ostream< \_CharT, \_Traits > & , const  
              std::normal\_distribution< \_RealType1 > & ) [friend]**

Inserts a normal\_distribution random number distribution \_\_x into the output stream \_\_os.

### **Parameters**

\_\_os An output stream.

\_\_x A normal\_distribution random number distribution.

## 5.613 `std::normal_distribution<_RealType>::param_type` Struct Reference 3039

### Returns

The output stream with the state of `__x` inserted or in an error state.

**5.612.5.2** `template<typename _RealType = double> template<typename  
_RealType1 > bool operator==( const std::normal_distribution<  
_RealType1 > & __d1, const std::normal_distribution< _RealType1  
> & __d2 ) [friend]`

Return true if two normal distributions have the same parameters and the sequences that would be generated are equal.

**5.612.5.3** `template<typename _RealType = double> template<typename  
_RealType1 , typename _CharT , typename _Traits >  
std::basic_istream<_CharT, _Traits>& operator>>  
( std::basic_istream< _CharT, _Traits > & ,  
std::normal_distribution< _RealType1 > & ) [friend]`

Extracts a `normal_distribution` random number distribution `__x` from the input stream `__is`.

### Parameters

`__is` An input stream.

`__x` A `normal_distribution` random number generator engine.

### Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.613 `std::normal_distribution<_RealType>::param_type` Struct Reference

### Public Types

- typedef [normal\\_distribution<\\_RealType>](#) `distribution_type`

**Public Member Functions**

- **param\_type** (`_RealType __mean=_RealType(0), _RealType __stddev=_RealType(1)`)
- `_RealType` **mean** () const
- `_RealType` **stddev** () const

**Friends**

- `bool` **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

**5.613.1 Detailed Description**

**template<typename \_RealType = double> struct `std::normal_distribution<_RealType>::param_type`**

Parameter type.

Definition at line 1976 of file `random.h`.

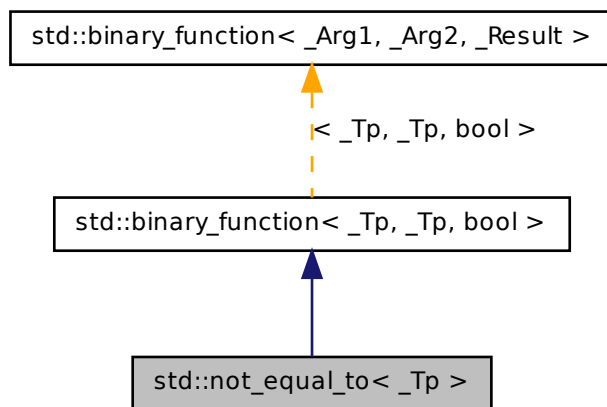
The documentation for this struct was generated from the following file:

- [random.h](#)

**5.614 `std::not_equal_to<_Tp>` Struct Template Reference**

One of the [comparison functors](#).

Inheritance diagram for std::not\_equal\_to< \_Tp >:



### Public Types

- typedef `_Tp` `first_argument_type`
- typedef `bool` `result_type`
- typedef `_Tp` `second_argument_type`

### Public Member Functions

- `bool operator() (const _Tp &__x, const _Tp &__y) const`

#### 5.614.1 Detailed Description

**template<typename \_Tp> struct std::not\_equal\_to< \_Tp >**

One of the [comparison functors](#).

Definition at line 214 of file `stl_function.h`.

### 5.614.2 Member Typedef Documentation

#### 5.614.2.1 `typedef _Tp std::binary_function<_Tp, _Tp, bool>::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

Definition at line 118 of file `stl_function.h`.

#### 5.614.2.2 `typedef bool std::binary_function<_Tp, _Tp, bool>::result_type` [inherited]

`result_type` is the return type

Definition at line 124 of file `stl_function.h`.

#### 5.614.2.3 `typedef _Tp std::binary_function<_Tp, _Tp, bool>::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

Definition at line 121 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

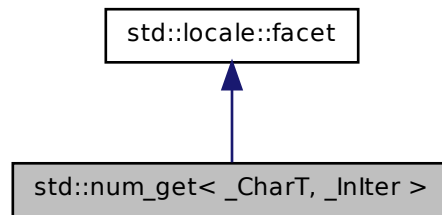
- [stl\\_function.h](#)

### 5.615 `std::num_get<_CharT, _InIter>` Class Template Reference

Primary class template [num\\_get](#).

This facet encapsulates the code to parse and return a number from a string. It is used by the istream numeric extraction operators.

Inheritance diagram for `std::num_get< _CharT, _InIter >`:



### Public Types

- typedef `_CharT` `char_type`
- typedef `_InIter` `iter_type`

### Public Member Functions

- `num_get` (`size_t` `__refs=0`)
- `iter_type get` (`iter_type` `__in`, `iter_type` `__end`, `ios_base &__io`, `ios_base::iostate &__err`, `void *&__v`) `const`
- `iter_type get` (`iter_type` `__in`, `iter_type` `__end`, `ios_base &__io`, `ios_base::iostate &__err`, `bool &__v`) `const`
- `iter_type get` (`iter_type` `__in`, `iter_type` `__end`, `ios_base &__io`, `ios_base::iostate &__err`, `long &__v`) `const`
- `iter_type get` (`iter_type` `__in`, `iter_type` `__end`, `ios_base &__io`, `ios_base::iostate &__err`, `unsigned short &__v`) `const`
- `iter_type get` (`iter_type` `__in`, `iter_type` `__end`, `ios_base &__io`, `ios_base::iostate &__err`, `unsigned int &__v`) `const`
- `iter_type get` (`iter_type` `__in`, `iter_type` `__end`, `ios_base &__io`, `ios_base::iostate &__err`, `unsigned long &__v`) `const`
- `iter_type get` (`iter_type` `__in`, `iter_type` `__end`, `ios_base &__io`, `ios_base::iostate &__err`, `long long &__v`) `const`
- `iter_type get` (`iter_type` `__in`, `iter_type` `__end`, `ios_base &__io`, `ios_base::iostate &__err`, `unsigned long long &__v`) `const`
- `iter_type get` (`iter_type` `__in`, `iter_type` `__end`, `ios_base &__io`, `ios_base::iostate &__err`, `float &__v`) `const`

- [iter\\_type get](#) (iter\_type \_\_in, iter\_type \_\_end, ios\_base &\_\_io, ios\_base::iostate &\_\_err, double &\_\_v) const
- [iter\\_type get](#) (iter\_type \_\_in, iter\_type \_\_end, ios\_base &\_\_io, ios\_base::iostate &\_\_err, long double &\_\_v) const

### Static Public Attributes

- static [locale::id](#) id

### Protected Member Functions

- virtual [~num\\_get](#) ()
- [iter\\_type \\_M\\_extract\\_float](#) (iter\_type, iter\_type, ios\_base &, ios\_base::iostate &, string &) const
- template<typename \_ValueT >  
[iter\\_type \\_M\\_extract\\_int](#) (iter\_type, iter\_type, ios\_base &, ios\_base::iostate &, \_ValueT &) const
- template<typename \_CharT2 >  
[\\_\\_gnu\\_cxx::\\_\\_enable\\_if< \\_\\_is\\_char< \\_CharT2 >::\\_\\_value, int >::\\_\\_type \\_M\\_find](#) (const \_CharT2 \*, size\_t \_\_len, \_CharT2 \_\_c) const
- template<typename \_CharT2 >  
[\\_\\_gnu\\_cxx::\\_\\_enable\\_if< !\\_\\_is\\_char< \\_CharT2 >::\\_\\_value, int >::\\_\\_type \\_M\\_find](#) (const \_CharT2 \* \_\_zero, size\_t \_\_len, \_CharT2 \_\_c) const
- virtual [iter\\_type do\\_get](#) (iter\_type, iter\_type, ios\_base &, ios\_base::iostate &, bool &) const
- virtual [iter\\_type do\\_get](#) (iter\_type \_\_beg, iter\_type \_\_end, ios\_base &\_\_io, ios\_base::iostate &\_\_err, long &\_\_v) const
- virtual [iter\\_type do\\_get](#) (iter\_type \_\_beg, iter\_type \_\_end, ios\_base &\_\_io, ios\_base::iostate &\_\_err, unsigned short &\_\_v) const
- virtual [iter\\_type do\\_get](#) (iter\_type \_\_beg, iter\_type \_\_end, ios\_base &\_\_io, ios\_base::iostate &\_\_err, unsigned int &\_\_v) const
- virtual [iter\\_type do\\_get](#) (iter\_type \_\_beg, iter\_type \_\_end, ios\_base &\_\_io, ios\_base::iostate &\_\_err, unsigned long &\_\_v) const
- virtual [iter\\_type do\\_get](#) (iter\_type \_\_beg, iter\_type \_\_end, ios\_base &\_\_io, ios\_base::iostate &\_\_err, long long &\_\_v) const
- virtual [iter\\_type do\\_get](#) (iter\_type \_\_beg, iter\_type \_\_end, ios\_base &\_\_io, ios\_base::iostate &\_\_err, unsigned long long &\_\_v) const
- virtual [iter\\_type do\\_get](#) (iter\_type, iter\_type, ios\_base &, ios\_base::iostate &\_\_err, float &) const
- virtual [iter\\_type do\\_get](#) (iter\_type, iter\_type, ios\_base &, ios\_base::iostate &\_\_err, double &) const
- virtual [iter\\_type do\\_get](#) (iter\_type, iter\_type, ios\_base &, ios\_base::iostate &\_\_err, long double &) const
- virtual [iter\\_type do\\_get](#) (iter\_type, iter\_type, ios\_base &, ios\_base::iostate &\_\_err, void \*&) const

### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static const char \* `_S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

### Friends

- class `locale::_Impl`

#### 5.615.1 Detailed Description

`template<typename _CharT, typename _InIter> class std::num_get<_CharT, _InIter>`

Primary class template [num\\_get](#).

This facet encapsulates the code to parse and return a number from a string. It is used by the istream numeric extraction operators. The [num\\_get](#) template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the [num\\_get](#) facet.

Definition at line 1916 of file `locale_facets.h`.

#### 5.615.2 Member Typedef Documentation

**5.615.2.1** `template<typename _CharT, typename _InIter> typedef _CharT std::num_get<_CharT, _InIter>::char_type`

Public typedefs.

Definition at line 1922 of file `locale_facets.h`.

**5.615.2.2** `template<typename _CharT, typename _InIter> typedef _InIter std::num_get<_CharT, _InIter>::iter_type`



Public typedefs.

Definition at line 1923 of file locale\_facets.h.

### 5.615.3 Constructor & Destructor Documentation

**5.615.3.1** `template<typename _CharT, typename _InIter > std::num_get<  
_CharT, _InIter >::num_get ( size_t __refs = 0 ) [inline,  
explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

#### Parameters

*refs* Passed to the base facet class.

Definition at line 1937 of file locale\_facets.h.

**5.615.3.2** `template<typename _CharT, typename _InIter > virtual  
std::num_get< _CharT, _InIter >::~~num_get ( ) [inline,  
protected, virtual]`

Destructor.

Definition at line 2106 of file locale\_facets.h.

### 5.615.4 Member Function Documentation

**5.615.4.1** `template<typename _CharT, typename _InIter > _InIter  
std::num_get< _CharT, _InIter >::do_get ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err, float &  
__v ) const [protected, virtual]`

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

#### See also

[get\(\)](#) for more details.

**Parameters**

*in* Start of input stream.  
*end* End of input stream.  
*io* Source of locale and flags.  
*err* Error flags to set.  
*v* Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 688 of file `locale_facets.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc >::c_str()`, `std::ios_base::eofbit`, and `std::basic_string<_CharT, _Traits, _Alloc >::reserve()`.

**5.615.4.2** `template<typename _CharT, typename _InIter > _InIter  
std::num_get<_CharT, _InIter >::do_get ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err, bool &  
__v ) const [protected, virtual]`

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

**See also**

[get\(\)](#) for more details.

**Parameters**

*in* Start of input stream.  
*end* End of input stream.  
*io* Source of locale and flags.  
*err* Error flags to set.  
*v* Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 592 of file `locale_facets.tcc`.

References `std::ios_base::_M_getloc()`, `std::ios_base::boolalpha`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::flags()`, and `std::ios_base::goodbit`.

Referenced by `std::num_get<_CharT, _InIter >::get()`.

**5.615.4.3** `template<typename _CharT, typename _InIter > virtual iter_type  
std::num_get< _CharT, _InIter >::do_get ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err, long &  
__v ) const [inline, protected, virtual]`

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

#### See also

[get\(\)](#) for more details.

#### Parameters

- in* Start of input stream.
- end* End of input stream.
- io* Source of locale and flags.
- err* Error flags to set.
- v* Value to format and insert.

#### Returns

Iterator after reading.

Definition at line 2174 of file locale\_facets.h.

**5.615.4.4** `template<typename _CharT, typename _InIter > virtual iter_type  
std::num_get< _CharT, _InIter >::do_get ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err,  
unsigned short & __v ) const [inline, protected,  
virtual]`

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

#### See also

[get\(\)](#) for more details.

**Parameters**

- in* Start of input stream.
- end* End of input stream.
- io* Source of locale and flags.
- err* Error flags to set.
- v* Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 2179 of file `locale_facets.h`.

```
5.615.4.5 template<typename _CharT, typename _InIter > virtual iter_type
std::num_get<_CharT, _InIter >::do_get (iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err,
unsigned int & __v) const [inline, protected, virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

**See also**

[get\(\)](#) for more details.

**Parameters**

- in* Start of input stream.
- end* End of input stream.
- io* Source of locale and flags.
- err* Error flags to set.
- v* Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 2184 of file `locale_facets.h`.

**5.615.4.6** `template<typename _CharT, typename _InIter > virtual iter_type  
std::num_get< _CharT, _InIter >::do_get ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err,  
unsigned long & __v ) const [inline, protected, virtual]`

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

#### See also

[get\(\)](#) for more details.

#### Parameters

- in* Start of input stream.
- end* End of input stream.
- io* Source of locale and flags.
- err* Error flags to set.
- v* Value to format and insert.

#### Returns

Iterator after reading.

Definition at line 2189 of file locale\_facets.h.

**5.615.4.7** `template<typename _CharT, typename _InIter > virtual iter_type  
std::num_get< _CharT, _InIter >::do_get ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err, long  
long & __v ) const [inline, protected, virtual]`

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

#### See also

[get\(\)](#) for more details.

#### Parameters

- in* Start of input stream.

*end* End of input stream.  
*io* Source of locale and flags.  
*err* Error flags to set.  
*v* Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 2195 of file `locale_facets.h`.

**5.615.4.8** `template<typename _CharT, typename _InIter > virtual iter_type  
std::num_get<_CharT, _InIter >::do_get ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err,  
unsigned long long & __v ) const [inline, protected,  
virtual]`

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

**See also**

[get\(\)](#) for more details.

**Parameters**

*in* Start of input stream.  
*end* End of input stream.  
*io* Source of locale and flags.  
*err* Error flags to set.  
*v* Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 2200 of file `locale_facets.h`.

**5.615.4.9** `template<typename _CharT, typename _InIter > _InIter  
std::num_get< _CharT, _InIter >::do_get ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err, long  
double & __v ) const [protected, virtual]`

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

#### See also

[get\(\)](#) for more details.

#### Parameters

*in* Start of input stream.  
*end* End of input stream.  
*io* Source of locale and flags.  
*err* Error flags to set.  
*v* Value to format and insert.

#### Returns

Iterator after reading.

Definition at line 735 of file locale\_facets.tcc.

References `std::basic_string< _CharT, _Traits, _Alloc >::c_str()`, `std::ios_base::eofbit`, and `std::basic_string< _CharT, _Traits, _Alloc >::reserve()`.

**5.615.4.10** `template<typename _CharT, typename _InIter > _InIter  
std::num_get< _CharT, _InIter >::do_get ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err,  
double & __v ) const [protected, virtual]`

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

#### See also

[get\(\)](#) for more details.

**Parameters**

*in* Start of input stream.  
*end* End of input stream.  
*io* Source of locale and flags.  
*err* Error flags to set.  
*v* Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 703 of file `locale_facets.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc >::c_str()`, `std::ios_base::eofbit`, and `std::basic_string<_CharT, _Traits, _Alloc >::reserve()`.

**5.615.4.11** `template<typename _CharT, typename _InIter > _InIter  
std::num_get<_CharT, _InIter >::do_get ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err, void  
*& __v ) const [protected, virtual]`

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

**See also**

[get\(\)](#) for more details.

**Parameters**

*in* Start of input stream.  
*end* End of input stream.  
*io* Source of locale and flags.  
*err* Error flags to set.  
*v* Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 750 of file `locale_facets.tcc`.

References `std::ios_base::basefield`, `std::ios_base::flags()`, and `std::ios_base::hex`.



**5.615.4.12** `template<typename _CharT, typename _InIter > iter_type  
std::num_get< _CharT, _InIter >::get ( iter_type __in, iter_type  
__end, ios_base & __io, ios_base::iostate & __err, float & __v )  
const [inline]`

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling [num\\_get::do\\_get\(\)](#).

The input characters are parsed like the scanf *g* specifier. The matching type length modifier is also used.

The decimal point character used is [numpunct::decimal\\_point\(\)](#). Digit grouping is interpreted according to [numpunct::grouping\(\)](#) and [numpunct::thousands\\_sep\(\)](#). If the pattern of digit groups isn't consistent, sets *err* to [ios\\_base::failbit](#).

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets *err* to [ios\\_base::failbit](#) and leaves *v* unaltered. Sets *err* to [ios\\_base::eofbit](#) if the stream is emptied.

#### Parameters

*in* Start of input stream.

*end* End of input stream.

*io* Source of locale and flags.

*err* Error flags to set.

*v* Value to format and insert.

#### Returns

Iterator after reading.

Definition at line 2058 of file locale\_facets.h.

References [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#).

**5.615.4.13** `template<typename _CharT, typename _InIter > iter_type  
std::num_get< _CharT, _InIter >::get ( iter_type __in, iter_type  
__end, ios_base & __io, ios_base::iostate & __err, double & __v )  
const [inline]`

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling [num\\_get::do\\_get\(\)](#).

The input characters are parsed like the `scanf g` specifier. The matching type length modifier is also used.

The decimal point character used is `num_punct::decimal_point()`. Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for `v`, `v` is set. Otherwise, sets `err` to `ios_base::failbit` and leaves `v` unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

### Parameters

- in* Start of input stream.
- end* End of input stream.
- io* Source of locale and flags.
- err* Error flags to set.
- v* Value to format and insert.

### Returns

Iterator after reading.

Definition at line 2063 of file `locale_facets.h`.

References `std::num_get< _CharT, _InIter >::do_get()`.

```
5.615.4.14 template<typename _CharT, typename _InIter> iter_type
std::num_get< _CharT, _InIter >::get (iter_type __in, iter_type
__end, ios_base & __io, ios_base::iostate & __err, void *& __v)
const [inline]
```

Numeric parsing.

Parses the input stream into the pointer variable `v`. It does so by calling `num_get::do_get()`.

The input characters are parsed like the `scanf p` specifier.

Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

Note that the digit grouping effect for pointers is a bit ambiguous in the standard and shouldn't be relied on. See DR 344.

If parsing the string yields a valid value for `v`, `v` is set. Otherwise, sets `err` to `ios_base::failbit` and leaves `v` unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

**Parameters**

- in* Start of input stream.
- end* End of input stream.
- io* Source of locale and flags.
- err* Error flags to set.
- v* Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 2100 of file locale\_facets.h.

References `std::num_get< _CharT, _InIter >::do_get()`.

```
5.615.4.15 template<typename _CharT, typename _InIter > iter_type
std::num_get< _CharT, _InIter >::get (iter_type __in, iter_type
__end, ios_base & __io, ios_base::iostate & __err, unsigned long
& __v) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags() & ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf` `o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets `err` to `ios_base::failbit` and leaves *v* unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

**Parameters**

- in* Start of input stream.
- end* End of input stream.
- io* Source of locale and flags.

*err* Error flags to set.

*v* Value to format and insert.

### Returns

Iterator after reading.

Definition at line 2014 of file locale\_facets.h.

References std::num\_get< \_CharT, \_InIter >::do\_get().

```
5.615.4.16 template<typename _CharT, typename _InIter> iter_type
std::num_get< _CharT, _InIter >::get (iter_type __in, iter_type
__end, ios_base & __io, ios_base::iostate & __err, bool & __v)
const [inline]
```

Numeric parsing.

Parses the input stream into the bool *v*. It does so by calling num\_get::do\_get().

If ios\_base::boolalpha is set, attempts to read ctype<CharT>::truename() or ctype<CharT>::falsename(). Sets *v* to true or false if successful. Sets *err* to ios\_base::failbit if reading the string fails. Sets *err* to ios\_base::eofbit if the stream is emptied.

If ios\_base::boolalpha is not set, proceeds as with reading a long, except if the value is 1, sets *v* to true, if the value is 0, sets *v* to false, and otherwise set *err* to ios\_base::failbit.

### Parameters

*in* Start of input stream.

*end* End of input stream.

*io* Source of locale and flags.

*err* Error flags to set.

*v* Value to format and insert.

### Returns

Iterator after reading.

Definition at line 1963 of file locale\_facets.h.

References std::num\_get< \_CharT, \_InIter >::do\_get().

Referenced by std::basic\_istream< \_CharT, \_Traits >::operator>>().

**5.615.4.17** `template<typename _CharT, typename _InIter > iter_type  
std::num_get< _CharT, _InIter >::get ( iter_type __in, iter_type  
__end, ios_base & __io, ios_base::iostate & __err, unsigned short  
& __v ) const [inline]`

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling [num\\_get::do\\_get\(\)](#).

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of *io.flags()* & [ios\\_base::basefield](#). If equal to [ios\\_base::oct](#), parses like the scanf o specifier. Else if equal to [ios\\_base::hex](#), parses like X specifier. Else if basefield equal to 0, parses like the i specifier. Otherwise, parses like d for signed and u for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to [num\\_punct::grouping\(\)](#) and [num\\_punct::thousands\\_sep\(\)](#). If the pattern of digit groups isn't consistent, sets err to [ios\\_base::failbit](#).

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to [ios\\_base::failbit](#) and leaves *v* unaltered. Sets err to [ios\\_base::eofbit](#) if the stream is emptied.

#### Parameters

- in* Start of input stream.
- end* End of input stream.
- io* Source of locale and flags.
- err* Error flags to set.
- v* Value to format and insert.

#### Returns

Iterator after reading.

Definition at line 2004 of file locale\_facets.h.

References [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#).

**5.615.4.18** `template<typename _CharT, typename _InIter > iter_type  
std::num_get< _CharT, _InIter >::get ( iter_type __in, iter_type  
__end, ios_base & __io, ios_base::iostate & __err, unsigned long  
long & __v ) const [inline]`

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags() & ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf` `o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets `err` to `ios_base::failbit` and leaves *v* unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

#### Parameters

- in* Start of input stream.
- end* End of input stream.
- io* Source of locale and flags.
- err* Error flags to set.
- v* Value to format and insert.

#### Returns

Iterator after reading.

Definition at line 2025 of file `locale_facets.h`.

References `std::num_get< _CharT, _InIter >::do_get()`.

```
5.615.4.19 template<typename _CharT, typename _InIter> iter_type
std::num_get< _CharT, _InIter >::get (iter_type __in, iter_type
__end, ios_base & __io, ios_base::iostate & __err, unsigned int &
__v) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf` `o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for `v`, `v` is set. Otherwise, sets `err` to `ios_base::failbit` and leaves `v` unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

#### Parameters

- in* Start of input stream.
- end* End of input stream.
- io* Source of locale and flags.
- err* Error flags to set.
- v* Value to format and insert.

#### Returns

Iterator after reading.

Definition at line 2009 of file `locale_facets.h`.

References `std::num_get< _CharT, _InIter >::do_get()`.

```
5.615.4.20 template<typename _CharT, typename _InIter > iter_type
std::num_get< _CharT, _InIter >::get (iter_type __in, iter_type
__end, ios_base & __io, ios_base::iostate & __err, long double &
__v) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable `v`. It does so by calling `num_get::do_get()`.

The input characters are parsed like the `scanf` `g` specifier. The matching type length modifier is also used.

The decimal point character used is `num_punct::decimal_point()`. Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for `v`, `v` is set. Otherwise, sets `err` to `ios_base::failbit` and leaves `v` unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

**Parameters**

- in* Start of input stream.
- end* End of input stream.
- io* Source of locale and flags.
- err* Error flags to set.
- v* Value to format and insert.

**Returns**

Iterator after reading.

Definition at line 2068 of file `locale_facets.h`.

References `std::num_get<_CharT, _InIter >::do_get()`.

```
5.615.4.21 template<typename _CharT, typename _InIter > iter_type
std::num_get<_CharT, _InIter >::get (iter_type __in, iter_type
__end, ios_base & __io, ios_base::iostate & __err, long long & __v
) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags() & ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf` `o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets `err` to `ios_base::failbit` and leaves *v* unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

**Parameters**

- in* Start of input stream.
- end* End of input stream.
- io* Source of locale and flags.



*err* Error flags to set.

*v* Value to format and insert.

### Returns

Iterator after reading.

Definition at line 2020 of file `locale_facets.h`.

References `std::num_get< _CharT, _InIter >::do_get()`.

```
5.615.4.22 template<typename _CharT, typename _InIter > iter_type
std::num_get< _CharT, _InIter >::get (iter_type __in, iter_type
__end, ios_base & __io, ios_base::iostate & __err, long & __v)
const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling [num\\_get::do\\_get\(\)](#).

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags()` & [ios\\_base::basefield](#). If equal to [ios\\_base::oct](#), parses like the `scanf` `o` specifier. Else if equal to [ios\\_base::hex](#), parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to [num\\_punct::grouping\(\)](#) and [num\\_punct::thousands\\_sep\(\)](#). If the pattern of digit groups isn't consistent, sets `err` to [ios\\_base::failbit](#).

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets `err` to [ios\\_base::failbit](#) and leaves *v* unaltered. Sets `err` to [ios\\_base::eofbit](#) if the stream is emptied.

### Parameters

*in* Start of input stream.

*end* End of input stream.

*io* Source of locale and flags.

*err* Error flags to set.

*v* Value to format and insert.

### Returns

Iterator after reading.

Definition at line 1999 of file `locale_facets.h`.

References `std::num_get< _CharT, _InIter >::do_get()`.

### 5.615.5 Member Data Documentation

#### 5.615.5.1 `template<typename _CharT, typename _InIter > locale::id` `std::num_get< _CharT, _InIter >::id` [static]

Numpunct facet id.

Definition at line 1927 of file `locale_facets.h`.

The documentation for this class was generated from the following files:

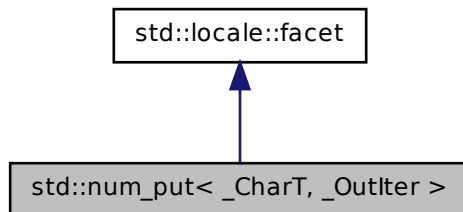
- [locale\\_facets.h](#)
- [locale\\_facets.tcc](#)

### 5.616 `std::num_put< _CharT, _OutIter >` Class Template Reference

Primary class template [num\\_put](#).

This facet encapsulates the code to convert a number to a string. It is used by the ostream numeric insertion operators.

Inheritance diagram for `std::num_put< _CharT, _OutIter >`:



## Public Types

- typedef `_CharT` `char_type`
- typedef `_OutIter` `iter_type`

## Public Member Functions

- `num_put` (`size_t` \_\_refs=0)
- `iter_type put` (`iter_type` \_\_s, `ios_base` &\_\_f, `char_type` \_\_fill, `const void *`\_\_v) `const`
- `iter_type put` (`iter_type` \_\_s, `ios_base` &\_\_f, `char_type` \_\_fill, `bool` \_\_v) `const`
- `iter_type put` (`iter_type` \_\_s, `ios_base` &\_\_f, `char_type` \_\_fill, `long` \_\_v) `const`
- `iter_type put` (`iter_type` \_\_s, `ios_base` &\_\_f, `char_type` \_\_fill, `unsigned long` \_\_v) `const`
- `iter_type put` (`iter_type` \_\_s, `ios_base` &\_\_f, `char_type` \_\_fill, `long long` \_\_v) `const`
- `iter_type put` (`iter_type` \_\_s, `ios_base` &\_\_f, `char_type` \_\_fill, `unsigned long long` \_\_v) `const`
- `iter_type put` (`iter_type` \_\_s, `ios_base` &\_\_f, `char_type` \_\_fill, `double` \_\_v) `const`
- `iter_type put` (`iter_type` \_\_s, `ios_base` &\_\_f, `char_type` \_\_fill, `long double` \_\_v) `const`

## Static Public Attributes

- static `locale::id` `id`

## Protected Member Functions

- virtual `~num_put` ()
- `void _M_group_float` (`const char *`\_\_grouping, `size_t` \_\_grouping\_size, `char_type` \_\_sep, `const char_type *`\_\_p, `char_type *`\_\_new, `char_type *`\_\_cs, `int` &\_\_len) `const`
- `void _M_group_int` (`const char *`\_\_grouping, `size_t` \_\_grouping\_size, `char_type` \_\_sep, `ios_base` &\_\_io, `char_type *`\_\_new, `char_type *`\_\_cs, `int` &\_\_len) `const`
- `template<typename _ValueT >`  
`iter_type _M_insert_float` (`iter_type`, `ios_base` &\_\_io, `char_type` \_\_fill, `char` \_\_mod, `_ValueT` \_\_v) `const`
- `template<typename _ValueT >`  
`iter_type _M_insert_int` (`iter_type`, `ios_base` &\_\_io, `char_type` \_\_fill, `_ValueT` \_\_v) `const`

- `void M_pad(char_type __fill, streamsize __w, ios_base &__io, char_type *__new, const char_type *__cs, int &__len) const`
- `virtual iter_type do_put(iter_type, ios_base &, char_type __fill, bool __v) const`
- `virtual iter_type do_put(iter_type __s, ios_base &__io, char_type __fill, long __v) const`
- `virtual iter_type do_put(iter_type __s, ios_base &__io, char_type __fill, unsigned long __v) const`
- `virtual iter_type do_put(iter_type __s, ios_base &__io, char_type __fill, long long __v) const`
- `virtual iter_type do_put(iter_type __s, ios_base &__io, char_type __fill, unsigned long long __v) const`
- `virtual iter_type do_put(iter_type, ios_base &, char_type __fill, double __v) const`
- `virtual iter_type do_put(iter_type, ios_base &, char_type __fill, long double __v) const`
- `virtual iter_type do_put(iter_type, ios_base &, char_type __fill, const void *__v) const`

#### Static Protected Member Functions

- `static __c_locale _S_clone_c_locale(__c_locale &__cloc) throw ()`
- `static void _S_create_c_locale(__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- `static void _S_destroy_c_locale(__c_locale &__cloc)`
- `static __c_locale _S_get_c_locale ()`
- `static const char * _S_get_c_name () throw ()`
- `static __c_locale _S_lc_ctype_c_locale(__c_locale __cloc, const char *__s)`

#### Friends

- `class locale::Impl`

#### 5.616.1 Detailed Description

`template<typename _CharT, typename _OutIter> class std::num_put<_CharT, _OutIter>`

Primary class template `num_put`.

This facet encapsulates the code to convert a number to a string. It is used by the ostream numeric insertion operators. The `num_put` template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual

functions. These virtual functions are hooks for developers to implement the behavior they require from the [num\\_put](#) facet.

Definition at line 2254 of file locale\_facets.h.

### **5.616.2 Member Typedef Documentation**

#### **5.616.2.1 template<typename \_CharT, typename \_OutIter > typedef \_CharT std::num\_put< \_CharT, \_OutIter >::char\_type**

Public typedefs.

Definition at line 2260 of file locale\_facets.h.

#### **5.616.2.2 template<typename \_CharT, typename \_OutIter > typedef \_OutIter std::num\_put< \_CharT, \_OutIter >::iter\_type**

Public typedefs.

Definition at line 2261 of file locale\_facets.h.

### **5.616.3 Constructor & Destructor Documentation**

#### **5.616.3.1 template<typename \_CharT, typename \_OutIter > std::num\_put< \_CharT, \_OutIter >::num\_put ( size\_t \_\_refs = 0 ) [inline, explicit]**

Constructor performs initialization.

This is the constructor provided by the standard.

#### **Parameters**

*refs* Passed to the base facet class.

Definition at line 2275 of file locale\_facets.h.

**5.616.3.2** `template<typename _CharT, typename _OutIter> virtual  
std::num_put<_CharT, _OutIter>::~~num_put( ) [inline,  
protected, virtual]`

Destructor.

Definition at line 2454 of file locale\_facets.h.

#### 5.616.4 Member Function Documentation

**5.616.4.1** `template<typename _CharT, typename _OutIter> virtual iter_type  
std::num_put<_CharT, _OutIter>::do_put( iter_type __s,  
ios_base & __io, char_type __fill, unsigned long __v ) const  
[inline, protected, virtual]`

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

##### Parameters

- s* Stream to write to.
- io* Source of locale and flags.
- fill* Char\_type to use for filling.
- v* Value to format and insert.

##### Returns

- Iterator after writing.

Definition at line 2478 of file locale\_facets.h.

**5.616.4.2** `template<typename _CharT, typename _OutIter> virtual  
iter_type std::num_put<_CharT, _OutIter>::do_put( iter_type  
__s, ios_base & __io, char_type __fill, long long __v ) const  
[inline, protected, virtual]`

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

**Parameters**

- s* Stream to write to.
- io* Source of locale and flags.
- fill* Char\_type to use for filling.
- v* Value to format and insert.

**Returns**

Iterator after writing.

Definition at line 2484 of file locale\_facets.h.

```
5.616.4.3 template<typename _CharT, typename _OutIter > virtual iter_type
std::num_put< _CharT, _OutIter >::do_put (iter_type __s,
ios_base & __io, char_type __fill, long __v) const [inline,
protected, virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

**Parameters**

- s* Stream to write to.
- io* Source of locale and flags.
- fill* Char\_type to use for filling.
- v* Value to format and insert.

**Returns**

Iterator after writing.

Definition at line 2474 of file locale\_facets.h.

```
5.616.4.4 template<typename _CharT, typename _OutIter > virtual iter_type
std::num_put< _CharT, _OutIter >::do_put (iter_type __s,
ios_base & __io, char_type __fill, unsigned long long __v) const
[inline, protected, virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

**Parameters**

- s* Stream to write to.
- io* Source of locale and flags.
- fill* Char\_type to use for filling.
- v* Value to format and insert.

**Returns**

Iterator after writing.

Definition at line 2489 of file locale\_facets.h.

```
5.616.4.5 template<typename _CharT, typename _OutIter > _OutIter
std::num_put< _CharT, _OutIter >::do_put (iter_type __s,
ios_base & __io, char_type __fill, long double __v) const
[protected, virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

**Parameters**

- s* Stream to write to.
- io* Source of locale and flags.
- fill* Char\_type to use for filling.
- v* Value to format and insert.

**Returns**

Iterator after writing.

Definition at line 1157 of file locale\_facets.tcc.

```
5.616.4.6 template<typename _CharT, typename _OutIter > _OutIter
std::num_put< _CharT, _OutIter >::do_put (iter_type __s,
ios_base & __io, char_type __fill, const void * __v) const
[protected, virtual]
```



Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

#### Parameters

- s* Stream to write to.
- io* Source of locale and flags.
- fill* Char\_type to use for filling.
- v* Value to format and insert.

#### Returns

Iterator after writing.

Definition at line 1164 of file locale\_facets.tcc.

References `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::uppercase`.

```
5.616.4.7 template<typename _CharT, typename _OutIter> _OutIter
std::num_put<_CharT, _OutIter>::do_put (iter_type
__s, ios_base & __io, char_type __fill, double __v) const
[protected, virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

#### Parameters

- s* Stream to write to.
- io* Source of locale and flags.
- fill* Char\_type to use for filling.
- v* Value to format and insert.

#### Returns

Iterator after writing.

Definition at line 1143 of file locale\_facets.tcc.

**5.616.4.8** `template<typename _CharT, typename _OutIter > _OutIter  
std::num_put< _CharT, _OutIter >::do_put ( iter_type __s,  
ios_base & __io, char_type __fill, bool __v ) const [protected,  
virtual]`

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

#### Parameters

- s* Stream to write to.
- io* Source of locale and flags.
- fill* Char\_type to use for filling.
- v* Value to format and insert.

#### Returns

Iterator after writing.

Definition at line 1091 of file locale\_facets.tcc.

References `std::ios_base::_M_getloc()`, `std::ios_base::adjustfield`, `std::ios_base::boolalpha`, `std::ios_base::flags()`, `std::ios_base::left`, and `std::ios_base::width()`.

Referenced by `std::num_put< _CharT, _OutIter >::put()`.

**5.616.4.9** `template<typename _CharT, typename _OutIter > iter_type  
std::num_put< _CharT, _OutIter >::put ( iter_type __s, ios_base &  
__f, char_type __fill, long __v ) const [inline]`

Numeric formatting.

Formats the integral value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, formats like the `printf` `o` specifier. Else if equal to `ios_base::hex`, formats like `x` or `X` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `d`, `ld`, `lld` for signed and `u`, `lu`, `llu` for unsigned values. Note that if both `oct` and `hex` are set, neither will take effect.

If [ios\\_base::showpos](#) is set, '+' is output before positive values. If [ios\\_base::showbase](#) is set, '0' precedes octal values (except 0) and '0[xX]' precedes hex values.

Thousands separators are inserted according to [numpunct::grouping\(\)](#) and [numpunct::thousands\\_sep\(\)](#). The decimal point character used is [numpunct::decimal\\_point\(\)](#).

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios\_base::adjustfield) == ios\_base::left`, result is padded at the end. If [ios\\_base::internal](#), then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

### Parameters

- s* Stream to write to.
- io* Source of locale and flags.
- fill* Char\_type to use for filling.
- v* Value to format and insert.

### Returns

Iterator after writing.

Definition at line 2335 of file locale\_facets.h.

References `std::num_put< _CharT, _OutIter >::do_put()`.

```
5.616.4.10 template<typename _CharT, typename _OutIter > iter_type
std::num_put< _CharT, _OutIter >::put (iter_type __s, ios_base
& __f, char_type __fill, long long __v) const [inline]
```

Numeric formatting.

Formats the integral value *v* and inserts it into a stream. It does so by calling [num\\_put::do\\_put\(\)](#).

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags() & ios\_base::basefield`. If equal to [ios\\_base::oct](#), formats like the printf o specifier. Else if equal to [ios\\_base::hex](#), formats like x or X with [ios\\_base::uppercase](#) unset or set respectively. Otherwise, formats like d, ld, lld for signed and u, lu, llu for unsigned values. Note that if both oct and hex are set, neither will take effect.

If [ios\\_base::showpos](#) is set, '+' is output before positive values. If [ios\\_base::showbase](#) is set, '0' precedes octal values (except 0) and '0[xX]' precedes hex values.

Thousands separators are inserted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. The decimal point character used is `num_punct::decimal_point()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

### Parameters

- s* Stream to write to.
- io* Source of locale and flags.
- fill* Char\_type to use for filling.
- v* Value to format and insert.

### Returns

Iterator after writing.

Definition at line 2345 of file locale\_facets.h.

References `std::num_put<_CharT, _OutIter>::do_put()`.

**5.616.4.11** `template<typename _CharT, typename _OutIter> iter_type  
std::num_put<_CharT, _OutIter>::put ( iter_type __s, ios_base  
& __f, char_type __fill, unsigned long __v ) const [inline]`

Numeric formatting.

Formats the integral value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags() & ios_base::basefield`. If equal to `ios_base::oct`, formats like the printf o specifier. Else if equal to `ios_base::hex`, formats like x or X with `ios_base::uppercase` unset or set respectively. Otherwise, formats like d, ld, lld for signed and u, lu, llu for unsigned values. Note that if both oct and hex are set, neither will take effect.

If `ios_base::showpos` is set, '+' is output before positive values. If `ios_base::showbase` is set, '0' precedes octal values (except 0) and '0[xX]' precedes hex values.

Thousands separators are inserted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. The decimal point character used is `num_punct::decimal_point()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

### Parameters

- s* Stream to write to.
- io* Source of locale and flags.
- fill* Char\_type to use for filling.
- v* Value to format and insert.

### Returns

Iterator after writing.

Definition at line 2339 of file locale\_facets.h.

References `std::num_put< _CharT, _OutIter >::do_put()`.

```
5.616.4.12 template<typename _CharT, typename _OutIter > iter_type
std::num_put< _CharT, _OutIter >::put (iter_type __s, ios_base
& __f, char_type __fill, unsigned long long __v) const
[inline]
```

Numeric formatting.

Formats the integral value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, formats like the printf o specifier. Else if equal to `ios_base::hex`, formats like x or X with `ios_base::uppercase` unset or set respectively. Otherwise, formats like d, ld, lld for signed and u, lu, llu for unsigned values. Note that if both oct and hex are set, neither will take effect.

If `ios_base::showpos` is set, '+' is output before positive values. If `ios_base::showbase` is set, '0' precedes octal values (except 0) and '0[xX]' precedes hex values.

Thousands separators are inserted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. The decimal point character used is `num_punct::decimal_point()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at

the end. If `ios_base::internal`, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

### Parameters

- s* Stream to write to.
- io* Source of locale and flags.
- fill* Char\_type to use for filling.
- v* Value to format and insert.

### Returns

Iterator after writing.

Definition at line 2349 of file `locale_facets.h`.

References `std::num_put<_CharT, _OutIter>::do_put()`.

**5.616.4.13** `template<typename _CharT, typename _OutIter> iter_type  
std::num_put<_CharT, _OutIter>::put ( iter_type __s, ios_base  
& __f, char_type __fill, double __v ) const [inline]`

Numeric formatting.

Formats the floating point value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags()` & `ios_base::floatfield`. If equal to `ios_base::fixed`, formats like the printf *f* specifier. Else if equal to `ios_base::scientific`, formats like *e* or *E* with `ios_base::uppercase` unset or set respectively. Otherwise, formats like *g* or *G* depending on uppercase. Note that if both fixed and scientific are set, the effect will also be like *g* or *G*.

The output precision is given by `io.precision()`. This precision is capped at `numeric_limits::digits10 + 2` (different for double and long double). The default precision is 6.

If `ios_base::showpos` is set, '+' is output before positive values. If `ios_base::showpoint` is set, a decimal point will always be output.

Thousands separators are inserted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. The decimal point character used is `num_punct::decimal_point()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at

the end. If [ios\\_base::internal](#), then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

### Parameters

- s* Stream to write to.
- io* Source of locale and flags.
- fill* Char\_type to use for filling.
- v* Value to format and insert.

### Returns

Iterator after writing.

Definition at line 2398 of file locale\_facets.h.

References [std::num\\_put< \\_CharT, \\_OutIter >::do\\_put\(\)](#).

**5.616.4.14** `template<typename _CharT, typename _OutIter > iter_type  
std::num_put< _CharT, _OutIter >::put ( iter_type __s, ios_base  
& __f, char_type __fill, const void * __v ) const [inline]`

Numeric formatting.

Formats the pointer value *v* and inserts it into a stream. It does so by calling [num\\_put::do\\_put\(\)](#).

This function formats *v* as an unsigned long with [ios\\_base::hex](#) and [ios\\_base::showbase](#) set.

### Parameters

- s* Stream to write to.
- io* Source of locale and flags.
- fill* Char\_type to use for filling.
- v* Value to format and insert.

### Returns

Iterator after writing.

Definition at line 2423 of file locale\_facets.h.

References [std::num\\_put< \\_CharT, \\_OutIter >::do\\_put\(\)](#).

**5.616.4.15** `template<typename _CharT, typename _OutIter > iter_type  
std::num_put< _CharT, _OutIter >::put ( iter_type __s, ios_base  
& __f, char_type __fill, long double __v ) const [inline]`

Numeric formatting.

Formats the floating point value *v* and inserts it into a stream. It does so by calling [num\\_put::do\\_put\(\)](#).

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags()` & [ios\\_base::floatfield](#). If equal to [ios\\_base::fixed](#), formats like the printf *f* specifier. Else if equal to [ios\\_base::scientific](#), formats like *e* or *E* with [ios\\_base::uppercase](#) unset or set respectively. Otherwise, formats like *g* or *G* depending on uppercase. Note that if both fixed and scientific are set, the effect will also be like *g* or *G*.

The output precision is given by `io.precision()`. This precision is capped at [numeric\\_limits::digits10](#) + 2 (different for double and long double). The default precision is 6.

If [ios\\_base::showpos](#) is set, '+' is output before positive values. If [ios\\_base::showpoint](#) is set, a decimal point will always be output.

Thousands separators are inserted according to [numpunct::grouping\(\)](#) and [numpunct::thousands\\_sep\(\)](#). The decimal point character used is [numpunct::decimal\\_point\(\)](#).

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios\_base::adjustfield) == ios\_base::left`, result is padded at the end. If [ios\\_base::internal](#), then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

### Parameters

- s* Stream to write to.
- io* Source of locale and flags.
- fill* Char\_type to use for filling.
- v* Value to format and insert.

### Returns

Iterator after writing.

Definition at line 2402 of file `locale_facets.h`.

References `std::num_put< _CharT, _OutIter >::do_put()`.



**5.616.4.16** `template<typename _CharT, typename _OutIter > iter_type  
std::num_put< _CharT, _OutIter >::put ( iter_type __s, ios_base  
& __f, char_type __fill, bool __v ) const [inline]`

Numeric formatting.

Formats the boolean *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

If `ios_base::boolalpha` is set, writes `ctype<CharT>::truename()` or `ctype<CharT>::falsename()`. Otherwise formats *v* as an int.

#### Parameters

- s* Stream to write to.
- io* Source of locale and flags.
- fill* `Char_type` to use for filling.
- v* Value to format and insert.

#### Returns

Iterator after writing.

Definition at line 2293 of file `locale_facets.h`.

References `std::num_put< _CharT, _OutIter >::do_put()`.

### 5.616.5 Member Data Documentation

**5.616.5.1** `template<typename _CharT, typename _OutIter > locale::id  
std::num_put< _CharT, _OutIter >::id [static]`

Numpunct facet id.

Definition at line 2265 of file `locale_facets.h`.

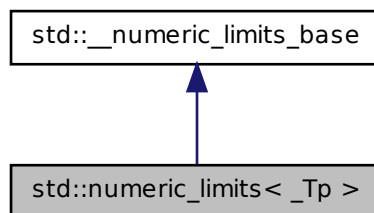
The documentation for this class was generated from the following files:

- [locale\\_facets.h](#)
- [locale\\_facets.tcc](#)

### 5.617 `std::numeric_limits< _Tp >` Struct Template Reference

Properties of fundamental types.

Inheritance diagram for `std::numeric_limits<_Tp>`:



#### Static Public Member Functions

- static constexpr `_Tp` [denorm\\_min](#) () throw ()
- static constexpr `_Tp` [epsilon](#) () throw ()
- static constexpr `_Tp` [infinity](#) () throw ()
- static constexpr `_Tp` [lowest](#) () throw ()
- static constexpr `_Tp` [max](#) () throw ()
- static constexpr `_Tp` [min](#) () throw ()
- static constexpr `_Tp` [quiet\\_NaN](#) () throw ()
- static constexpr `_Tp` [round\\_error](#) () throw ()
- static constexpr `_Tp` [signaling\\_NaN](#) () throw ()

#### Static Public Attributes

- static constexpr int [digits](#)
- static constexpr int [digits10](#)
- static constexpr `float_denorm_style` [has\\_denorm](#)
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool [has\\_infinity](#)
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool [is\\_bounded](#)
- static constexpr bool [is\\_exact](#)
- static constexpr bool [is\\_iec559](#)
- static constexpr bool [is\\_integer](#)
- static constexpr bool [is\\_modulo](#)

- static constexpr bool [is\\_signed](#)
- static constexpr bool [is\\_specialized](#)
- static constexpr int [max\\_digits10](#)
- static constexpr int [max\\_exponent](#)
- static constexpr int [max\\_exponent10](#)
- static constexpr int [min\\_exponent](#)
- static constexpr int [min\\_exponent10](#)
- static constexpr int [radix](#)
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool [traps](#)

### 5.617.1 Detailed Description

**template<typename \_Tp> struct std::numeric\_limits<\_Tp>**

Properties of fundamental types. This class allows a program to obtain information about the representation of a fundamental type on a given platform. For non-fundamental types, the functions will return 0 and the data members will all be `false`.

`_GLIBCXX_RESOLVE_LIB_DEFECTS`: DRs 201 and 184 (hi Gaby!) are noted, but not incorporated in this documented (yet).

Definition at line 304 of file limits.

### 5.617.2 Member Function Documentation

**5.617.2.1** **template<typename \_Tp> static constexpr \_Tp  
std::numeric\_limits<\_Tp>::denorm\_min ( ) throw () [inline,  
static]**

The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

Definition at line 349 of file limits.

**5.617.2.2** **template<typename \_Tp> static constexpr \_Tp  
std::numeric\_limits<\_Tp>::epsilon ( ) throw () [inline,  
static]**

The *machine epsilon*: the difference between 1 and the least value greater than 1 that is representable.

Definition at line 325 of file limits.

**5.617.2.3** `template<typename _Tp> static constexpr _Tp  
std::numeric_limits<_Tp>::infinity ( ) throw () [inline,  
static]`

The representation of positive infinity, if `has_infinity`.

Definition at line 333 of file `limits`.

**5.617.2.4** `template<typename _Tp> static constexpr _Tp  
std::numeric_limits<_Tp>::lowest ( ) throw () [inline,  
static]`

A finite value `x` such that there is no other finite value `y` where `y < x`.

Definition at line 319 of file `limits`.

**5.617.2.5** `template<typename _Tp> static constexpr _Tp  
std::numeric_limits<_Tp>::max ( ) throw () [inline,  
static]`

The maximum finite value.

Definition at line 313 of file `limits`.

**5.617.2.6** `template<typename _Tp> static constexpr _Tp  
std::numeric_limits<_Tp>::min ( ) throw () [inline,  
static]`

The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

Definition at line 309 of file `limits`.

**5.617.2.7** `template<typename _Tp> static constexpr _Tp  
std::numeric_limits<_Tp>::quiet_NaN ( ) throw () [inline,  
static]`

The representation of a quiet *Not a Number*, if `has_quiet_NaN`.

Definition at line 338 of file `limits`.

**5.617.2.8** `template<typename _Tp> static constexpr _Tp  
std::numeric_limits<_Tp>::round_error ( ) throw () [inline,  
static]`

The maximum rounding error measurement (see LIA-1).

Definition at line 329 of file limits.

**5.617.2.9** `template<typename _Tp> static constexpr _Tp  
std::numeric_limits<_Tp>::signaling_NaN ( ) throw ()  
[inline, static]`

The representation of a signaling *Not a Number*, if `has_signaling_NaN`.

Definition at line 343 of file limits.

### 5.617.3 Member Data Documentation

**5.617.3.1** `constexpr int std::__numeric_limits_base::digits [static,  
inherited]`

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

Definition at line 201 of file limits.

**5.617.3.2** `constexpr int std::__numeric_limits_base::digits10 [static,  
inherited]`

The number of base 10 digits that can be represented without change.

Definition at line 204 of file limits.

**5.617.3.3** `constexpr float_denorm_style std::__numeric_limits_base::has_-  
denorm [static, inherited]`

See [std::float\\_denorm\\_style](#) for more information.

Definition at line 258 of file limits.

**5.617.3.4 constexpr bool std::\_\_numeric\_limits\_base::has\_denorm\_loss**  
**[static, inherited]**

*True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result. [18.2.1.2]/42*

Definition at line 262 of file limits.

**5.617.3.5 constexpr bool std::\_\_numeric\_limits\_base::has\_infinity** **[static, inherited]**

True if the type has a representation for positive infinity.

Definition at line 247 of file limits.

**5.617.3.6 constexpr bool std::\_\_numeric\_limits\_base::has\_quiet\_NaN**  
**[static, inherited]**

True if the type has a representation for a quiet (non-signaling) *Not a Number*.

Definition at line 251 of file limits.

**5.617.3.7 constexpr bool std::\_\_numeric\_limits\_base::has\_signaling\_NaN**  
**[static, inherited]**

True if the type has a representation for a signaling *Not a Number*.

Definition at line 255 of file limits.

**5.617.3.8 constexpr bool std::\_\_numeric\_limits\_base::is\_bounded** **[static, inherited]**

*True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types. [18.2.1.2]/54*

Definition at line 271 of file limits.

**5.617.3.9 constexpr bool std::\_\_numeric\_limits\_base::is\_exact** **[static, inherited]**

True if the type uses an exact representation. *All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer. [18.2.1.2]/15*

Definition at line 223 of file limits.

**5.617.3.10 constexpr bool std::\_\_numeric\_limits\_base::is\_iec559 [static, inherited]**

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

Definition at line 266 of file limits.

**5.617.3.11 constexpr bool std::\_\_numeric\_limits\_base::is\_integer [static, inherited]**

True if the type is integer. Is this supposed to be *if the type is integral*?

Definition at line 217 of file limits.

**5.617.3.12 constexpr bool std::\_\_numeric\_limits\_base::is\_modulo [static, inherited]**

True if the type is *modulo*, that is, if it is possible to add two positive numbers and have a result that wraps around to a third number that is less. Typically false for floating types, true for unsigned integers, and true for signed integers.

Definition at line 277 of file limits.

**5.617.3.13 constexpr bool std::\_\_numeric\_limits\_base::is\_signed [static, inherited]**

True if the type is signed.

Definition at line 213 of file limits.

**5.617.3.14 constexpr bool std::\_\_numeric\_limits\_base::is\_specialized [static, inherited]**

This will be true for all fundamental types (which have specializations), and false for everything else.

Definition at line 196 of file limits.

**5.617.3.15 constexpr int std::\_\_numeric\_limits\_base::max\_digits10 [static, inherited]**

The number of base 10 digits required to ensure that values which differ are always differentiated.

Definition at line 209 of file limits.

**5.617.3.16** `constexpr int std::__numeric_limits_base::max_exponent`  
`[static, inherited]`

The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

Definition at line 240 of file limits.

**5.617.3.17** `constexpr int std::__numeric_limits_base::max_exponent10`  
`[static, inherited]`

The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

Definition at line 244 of file limits.

**5.617.3.18** `constexpr int std::__numeric_limits_base::min_exponent`  
`[static, inherited]`

The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

Definition at line 231 of file limits.

**5.617.3.19** `constexpr int std::__numeric_limits_base::min_exponent10`  
`[static, inherited]`

The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

Definition at line 235 of file limits.

**5.617.3.20** `constexpr int std::__numeric_limits_base::radix` `[static, inherited]`

For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

Definition at line 227 of file limits.



**5.617.3.21** `constexpr float_round_style std::__numeric_limits_base::round_style [static, inherited]`

See `std::float_round_style` for more information. This is only meaningful for floating types; integer types will all be `round_toward_zero`.

Definition at line 288 of file `limits`.

**5.617.3.22** `constexpr bool std::__numeric_limits_base::tinyness_before [static, inherited]`

True if tininess is detected before rounding. (see IEC 559)

Definition at line 283 of file `limits`.

**5.617.3.23** `constexpr bool std::__numeric_limits_base::traps [static, inherited]`

True if trapping is implemented for this type.

Definition at line 280 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

**5.618** `std::numeric_limits< bool >` Struct Template Reference

[numeric\\_limits<bool>](#) specialization.

**Static Public Member Functions**

- static constexpr bool **denorm\_min** () throw ()
- static constexpr bool **epsilon** () throw ()
- static constexpr bool **infinity** () throw ()
- static constexpr bool **lowest** () throw ()
- static constexpr bool **max** () throw ()
- static constexpr bool **min** () throw ()
- static constexpr bool **quiet\_NaN** () throw ()
- static constexpr bool **round\_error** () throw ()
- static constexpr bool **signaling\_NaN** () throw ()

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

#### 5.618.1 Detailed Description

`template<> struct std::numeric_limits< bool >`

[numeric\\_limits<bool>](#) specialization.

Definition at line 371 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

### 5.619 `std::numeric_limits< char >` Struct Template Reference

[numeric\\_limits<char>](#) specialization.

### Static Public Member Functions

- static constexpr char **denorm\_min** () throw ()
- static constexpr char **epsilon** () throw ()
- static constexpr char **infinity** () throw ()
- static constexpr char **lowest** () throw ()
- static constexpr char **max** () throw ()
- static constexpr char **min** () throw ()
- static constexpr char **quiet\_NaN** () throw ()
- static constexpr char **round\_error** () throw ()
- static constexpr char **signaling\_NaN** () throw ()

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

### 5.619.1 Detailed Description

`template<> struct std::numeric_limits< char >`

[numeric\\_limits<char>](#) specialization.

Definition at line 440 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.620 `std::numeric_limits< char16_t >` Struct Template Reference

[numeric\\_limits<char16\\_t>](#) specialization.

### Static Public Member Functions

- static constexpr `char16_t` **denorm\_min** () throw ()
- static constexpr `char16_t` **epsilon** () throw ()
- static constexpr `char16_t` **infinity** () throw ()
- static constexpr `char16_t` **lowest** () throw ()
- static constexpr `char16_t` **max** () throw ()
- static constexpr `char16_t` **min** () throw ()
- static constexpr `char16_t` **quiet\_NaN** () throw ()
- static constexpr `char16_t` **round\_error** () throw ()
- static constexpr `char16_t` **signaling\_NaN** () throw ()

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**

- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

#### 5.620.1 Detailed Description

`template<> struct std::numeric_limits< char16_t >`

[numeric\\_limits<char16\\_t>](#) specialization.

Definition at line 713 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

### 5.621 `std::numeric_limits< char32_t >` Struct Template Reference

[numeric\\_limits<char32\\_t>](#) specialization.

#### Static Public Member Functions

- static constexpr `char32_t` **denorm\_min** () throw ()
- static constexpr `char32_t` **epsilon** () throw ()
- static constexpr `char32_t` **infinity** () throw ()
- static constexpr `char32_t` **lowest** () throw ()
- static constexpr `char32_t` **max** () throw ()
- static constexpr `char32_t` **min** () throw ()
- static constexpr `char32_t` **quiet\_NaN** () throw ()
- static constexpr `char32_t` **round\_error** () throw ()
- static constexpr `char32_t` **signaling\_NaN** () throw ()

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

#### 5.621.1 Detailed Description

`template<> struct std::numeric_limits< char32_t >`

[numeric\\_limits<char32\\_t>](#) specialization.

Definition at line 783 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.622 `std::numeric_limits< double >` Struct Template Reference

[numeric\\_limits<double>](#) specialization.

### Static Public Member Functions

- static constexpr double **denorm\_min** () throw ()
- static constexpr double **epsilon** () throw ()
- static constexpr double **infinity** () throw ()
- static constexpr double **lowest** () throw ()
- static constexpr double **max** () throw ()
- static constexpr double **min** () throw ()
- static constexpr double **quiet\_NaN** () throw ()
- static constexpr double **round\_error** () throw ()
- static constexpr double **signaling\_NaN** () throw ()

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

### 5.622.1 Detailed Description

`template<> struct std::numeric_limits< double >`

[numeric\\_limits<double>](#) specialization.

Definition at line 1474 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.623 `std::numeric_limits< float >` Struct Template Reference

[numeric\\_limits<float>](#) specialization.

### Static Public Member Functions

- static constexpr float **denorm\_min** () throw ()
- static constexpr float **epsilon** () throw ()
- static constexpr float **infinity** () throw ()
- static constexpr float **lowest** () throw ()
- static constexpr float **max** () throw ()
- static constexpr float **min** () throw ()
- static constexpr float **quiet\_NaN** () throw ()
- static constexpr float **round\_error** () throw ()
- static constexpr float **signaling\_NaN** () throw ()

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**



- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

#### 5.623.1 Detailed Description

`template<> struct std::numeric_limits< float >`

[numeric\\_limits<float>](#) specialization.

Definition at line 1399 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.624 `std::numeric_limits< int >` Struct Template Reference

[numeric\\_limits<int>](#) specialization.

#### Static Public Member Functions

- static constexpr int **denorm\_min** () throw ()
- static constexpr int **epsilon** () throw ()
- static constexpr int **infinity** () throw ()
- static constexpr int **lowest** () throw ()
- static constexpr int **max** () throw ()
- static constexpr int **min** () throw ()
- static constexpr int **quiet\_NaN** () throw ()
- static constexpr int **round\_error** () throw ()
- static constexpr int **signaling\_NaN** () throw ()

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

#### 5.624.1 Detailed Description

`template<> struct std::numeric_limits< int >`

[numeric\\_limits<int>](#) specialization.

Definition at line 989 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

### 5.625 `std::numeric_limits< long >` Struct Template Reference

[numeric\\_limits<long>](#) specialization.

### Static Public Member Functions

- static constexpr long **denorm\_min** () throw ()
- static constexpr long **epsilon** () throw ()
- static constexpr long **infinity** () throw ()
- static constexpr long **lowest** () throw ()
- static constexpr long **max** () throw ()
- static constexpr long **min** () throw ()
- static constexpr long **quiet\_NaN** () throw ()
- static constexpr long **round\_error** () throw ()
- static constexpr long **signaling\_NaN** () throw ()

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

### 5.625.1 Detailed Description

`template<> struct std::numeric_limits< long >`

[numeric\\_limits<long>](#) specialization.

Definition at line 1125 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.626 `std::numeric_limits< long double >` Struct Template Reference

[numeric\\_limits<long double>](#) specialization.

### Static Public Member Functions

- static constexpr long double **denorm\_min** () throw ()
- static constexpr long double **epsilon** () throw ()
- static constexpr long double **infinity** () throw ()
- static constexpr long double **lowest** () throw ()
- static constexpr long double **max** () throw ()
- static constexpr long double **min** () throw ()
- static constexpr long double **quiet\_NaN** () throw ()
- static constexpr long double **round\_error** () throw ()
- static constexpr long double **signaling\_NaN** () throw ()

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**

- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

#### 5.626.1 Detailed Description

`template<> struct std::numeric_limits< long double >`

[numeric\\_limits<long double>](#) specialization.

Definition at line 1549 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.627 `std::numeric_limits< long long >` Struct Template Reference

[numeric\\_limits<long long>](#) specialization.

#### Static Public Member Functions

- static constexpr long long **denorm\_min** () throw ()
- static constexpr long long **epsilon** () throw ()
- static constexpr long long **infinity** () throw ()
- static constexpr long long **lowest** () throw ()
- static constexpr long long **max** () throw ()
- static constexpr long long **min** () throw ()
- static constexpr long long **quiet\_NaN** () throw ()
- static constexpr long long **round\_error** () throw ()
- static constexpr long long **signaling\_NaN** () throw ()

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

#### 5.627.1 Detailed Description

`template<> struct std::numeric_limits< long long >`

[numeric\\_limits<long long>](#) specialization.

Definition at line 1261 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

### 5.628 `std::numeric_limits< short >` Struct Template Reference

[numeric\\_limits<short>](#) specialization.

### Static Public Member Functions

- static constexpr short **denorm\_min** () throw ()
- static constexpr short **epsilon** () throw ()
- static constexpr short **infinity** () throw ()
- static constexpr short **lowest** () throw ()
- static constexpr short **max** () throw ()
- static constexpr short **min** () throw ()
- static constexpr short **quiet\_NaN** () throw ()
- static constexpr short **round\_error** () throw ()
- static constexpr short **signaling\_NaN** () throw ()

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

### 5.628.1 Detailed Description

`template<> struct std::numeric_limits< short >`

[numeric\\_limits<short>](#) specialization.

Definition at line 853 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.629 `std::numeric_limits< signed char >` Struct Template Reference

[numeric\\_limits<signed char>](#) specialization.

### Static Public Member Functions

- static constexpr signed char **denorm\_min** () throw ()
- static constexpr signed char **epsilon** () throw ()
- static constexpr signed char **infinity** () throw ()
- static constexpr signed char **lowest** () throw ()
- static constexpr signed char **max** () throw ()
- static constexpr signed char **min** () throw ()
- static constexpr signed char **quiet\_NaN** () throw ()
- static constexpr signed char **round\_error** () throw ()
- static constexpr signed char **signaling\_NaN** () throw ()

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**



- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

#### 5.629.1 Detailed Description

`template<> struct std::numeric_limits< signed char >`

[numeric\\_limits<signed char>](#) specialization.

Definition at line 507 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

### 5.630 `std::numeric_limits< unsigned char >` Struct Template Reference

[numeric\\_limits<unsigned char>](#) specialization.

#### Static Public Member Functions

- static constexpr unsigned char **denorm\_min** () throw ()
- static constexpr unsigned char **epsilon** () throw ()
- static constexpr unsigned char **infinity** () throw ()
- static constexpr unsigned char **lowest** () throw ()
- static constexpr unsigned char **max** () throw ()
- static constexpr unsigned char **min** () throw ()
- static constexpr unsigned char **quiet\_NaN** () throw ()
- static constexpr unsigned char **round\_error** () throw ()
- static constexpr unsigned char **signaling\_NaN** () throw ()

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

#### 5.630.1 Detailed Description

`template<> struct std::numeric_limits< unsigned char >`

[numeric\\_limits<unsigned char>](#) specialization.

Definition at line 575 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

### 5.631 `std::numeric_limits< unsigned int >` Struct Template Reference

[numeric\\_limits<unsigned int>](#) specialization.

### Static Public Member Functions

- static constexpr unsigned int **denorm\_min** () throw ()
- static constexpr unsigned int **epsilon** () throw ()
- static constexpr unsigned int **infinity** () throw ()
- static constexpr unsigned int **lowest** () throw ()
- static constexpr unsigned int **max** () throw ()
- static constexpr unsigned int **min** () throw ()
- static constexpr unsigned int **quiet\_NaN** () throw ()
- static constexpr unsigned int **round\_error** () throw ()
- static constexpr unsigned int **signaling\_NaN** () throw ()

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

### 5.631.1 Detailed Description

`template<> struct std::numeric_limits< unsigned int >`

[numeric\\_limits<unsigned int>](#) specialization.

Definition at line 1056 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.632 `std::numeric_limits< unsigned long >` Struct Template Reference

[numeric\\_limits<unsigned long>](#) specialization.

### Static Public Member Functions

- static constexpr unsigned long **denorm\_min** () throw ()
- static constexpr unsigned long **epsilon** () throw ()
- static constexpr unsigned long **infinity** () throw ()
- static constexpr unsigned long **lowest** () throw ()
- static constexpr unsigned long **max** () throw ()
- static constexpr unsigned long **min** () throw ()
- static constexpr unsigned long **quiet\_NaN** () throw ()
- static constexpr unsigned long **round\_error** () throw ()
- static constexpr unsigned long **signaling\_NaN** () throw ()

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**

### 5.633 `std::numeric_limits< unsigned long long >` Struct Template Reference

- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

#### 5.632.1 Detailed Description

`template<> struct std::numeric_limits< unsigned long >`

[numeric\\_limits<unsigned long>](#) specialization.

Definition at line 1192 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

### 5.633 `std::numeric_limits< unsigned long long >` Struct Template Reference

[numeric\\_limits<unsigned long long>](#) specialization.

#### Static Public Member Functions

- static constexpr unsigned long long **denorm\_min** () throw ()
- static constexpr unsigned long long **epsilon** () throw ()
- static constexpr unsigned long long **infinity** () throw ()
- static constexpr unsigned long long **lowest** () throw ()
- static constexpr unsigned long long **max** () throw ()
- static constexpr unsigned long long **min** () throw ()
- static constexpr unsigned long long **quiet\_NaN** () throw ()
- static constexpr unsigned long long **round\_error** () throw ()
- static constexpr unsigned long long **signaling\_NaN** () throw ()

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

#### 5.633.1 Detailed Description

`template<> struct std::numeric_limits< unsigned long long >`

[numeric\\_limits<unsigned long long>](#) specialization.

Definition at line 1330 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

### 5.634 `std::numeric_limits< unsigned short >` Struct Template Reference

[numeric\\_limits<unsigned short>](#) specialization.

### Static Public Member Functions

- static constexpr unsigned short **denorm\_min** () throw ()
- static constexpr unsigned short **epsilon** () throw ()
- static constexpr unsigned short **infinity** () throw ()
- static constexpr unsigned short **lowest** () throw ()
- static constexpr unsigned short **max** () throw ()
- static constexpr unsigned short **min** () throw ()
- static constexpr unsigned short **quiet\_NaN** () throw ()
- static constexpr unsigned short **round\_error** () throw ()
- static constexpr unsigned short **signaling\_NaN** () throw ()

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

### 5.634.1 Detailed Description

`template<> struct std::numeric_limits< unsigned short >`

[numeric\\_limits<unsigned short>](#) specialization.

Definition at line 920 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 5.635 `std::numeric_limits< wchar_t >` Struct Template Reference

[numeric\\_limits<wchar\\_t>](#) specialization.

### Static Public Member Functions

- static constexpr `wchar_t` **denorm\_min** () throw ()
- static constexpr `wchar_t` **epsilon** () throw ()
- static constexpr `wchar_t` **infinity** () throw ()
- static constexpr `wchar_t` **lowest** () throw ()
- static constexpr `wchar_t` **max** () throw ()
- static constexpr `wchar_t` **min** () throw ()
- static constexpr `wchar_t` **quiet\_NaN** () throw ()
- static constexpr `wchar_t` **round\_error** () throw ()
- static constexpr `wchar_t` **signaling\_NaN** () throw ()

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**



- static constexpr bool `is_specialized`
- static constexpr int `max_digits10`
- static constexpr int `max_exponent`
- static constexpr int `max_exponent10`
- static constexpr int `min_exponent`
- static constexpr int `min_exponent10`
- static constexpr int `radix`
- static constexpr [float\\_round\\_style](#) `round_style`
- static constexpr bool `tinyness_before`
- static constexpr bool `traps`

#### 5.635.1 Detailed Description

`template<> struct std::numeric_limits< wchar_t >`

[numeric\\_limits<wchar\\_t>](#) specialization.

Definition at line 644 of file `limits`.

The documentation for this struct was generated from the following file:

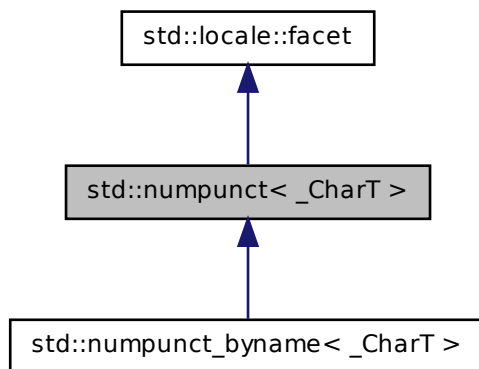
- [limits](#)

## 5.636 `std::numpunct<_CharT>` Class Template Reference

Primary class template `numpunct`.

This facet stores several pieces of information related to printing and scanning numbers, such as the decimal point character. It takes a template parameter specifying the char type. The `numpunct` facet is used by streams for many I/O operations involving numbers.

Inheritance diagram for `std::numpunct<_CharT>`:



### Public Types

- `typedef __numpunct_cache<_CharT> __cache_type`
- `typedef _CharT char_type`
- `typedef basic_string<_CharT> string_type`

### Public Member Functions

- `numpunct` (`size_t __refs=0`)
- `numpunct` (`__cache_type *__cache, size_t __refs=0`)
- `numpunct` (`__c_locale __cloc, size_t __refs=0`)
- `char_type decimal_point` () const
- `string_type falsename` () const
- `string grouping` () const
- `char_type thousands_sep` () const
- `string_type truename` () const

### Static Public Attributes

- static `locale::id id`

**Protected Member Functions**

- virtual [~numpunct](#) ()
- template<>  
void [\\_M\\_initialize\\_numpunct](#) (\_\_c\_locale \_\_cloc)
- template<>  
void [\\_M\\_initialize\\_numpunct](#) (\_\_c\_locale \_\_cloc)
- void [\\_M\\_initialize\\_numpunct](#) (\_\_c\_locale \_\_cloc=0)
- virtual [char\\_type do\\_decimal\\_point](#) () const
- virtual [string\\_type do\\_falsename](#) () const
- virtual [string do\\_grouping](#) () const
- virtual [char\\_type do\\_thousands\\_sep](#) () const
- virtual [string\\_type do\\_truename](#) () const

**Static Protected Member Functions**

- static \_\_c\_locale [\\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale [\\_S\\_get\\_c\\_locale](#) ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static \_\_c\_locale [\\_S\\_lc\\_ctype\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \*\_\_s)

**Protected Attributes**

- \_\_cache\_type \* [\\_M\\_data](#)

**Friends**

- class [locale::Impl](#)

**5.636.1 Detailed Description**

**template<typename \_CharT> class std::numpunct<\_CharT>**

Primary class template numpunct.

This facet stores several pieces of information related to printing and scanning numbers, such as the decimal point character. It takes a template parameter specifying the char type. The numpunct facet is used by streams for many I/O operations involving numbers. The numpunct template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These

virtual functions are hooks for developers to implement the behavior they require from a `numpunct` facet.

Definition at line 1642 of file `locale_facets.h`.

### 5.636.2 Member Typedef Documentation

#### 5.636.2.1 `template<typename _CharT> typedef _CharT std::numpunct<_CharT>::char_type`

Public typedefs.

Reimplemented in [std::numpunct\\_byname<\\_CharT>](#).

Definition at line 1648 of file `locale_facets.h`.

#### 5.636.2.2 `template<typename _CharT> typedef basic_string<_CharT> std::numpunct<_CharT>::string_type`

Public typedefs.

Reimplemented in [std::numpunct\\_byname<\\_CharT>](#).

Definition at line 1649 of file `locale_facets.h`.

### 5.636.3 Constructor & Destructor Documentation

#### 5.636.3.1 `template<typename _CharT> std::numpunct<_CharT>::numpunct( size_t __refs = 0 ) [inline, explicit]`

Numpunct constructor.

#### Parameters

*refs* Refcount to pass to the base class.

Definition at line 1666 of file `locale_facets.h`.

**5.636.3.2** `template<typename _CharT> std::num_punct<_CharT>  
>::num_punct( __cache_type * __cache, size_t __refs = 0 )  
[inline, explicit]`

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up the predefined locale facets.

#### Parameters

*cache* \_\_num\_punct\_cache object.

*refs* Refcount to pass to the base class.

Definition at line 1680 of file locale\_facets.h.

**5.636.3.3** `template<typename _CharT> std::num_punct<_CharT>  
>::num_punct( __c_locale __cloc, size_t __refs = 0 ) [inline,  
explicit]`

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

#### Parameters

*cloc* The C locale.

*refs* Refcount to pass to the base class.

Definition at line 1694 of file locale\_facets.h.

**5.636.3.4** `template<typename _CharT> virtual std::num_punct<_CharT>  
>::~num_punct( ) [protected, virtual]`

Destructor.

#### 5.636.4 Member Function Documentation

**5.636.4.1** `template<typename _CharT> char_type std::num_punct<_CharT>  
>::decimal_point( ) const [inline]`

Return decimal point character.

This function returns a `char_type` to use as a decimal point. It does so by returning returning `numpunct<char_type>::do_decimal_point()`.

#### Returns

*char\_type* representing a decimal point.

Definition at line 1708 of file `locale_facets.h`.

References `std::numpunct<_CharT>::do_decimal_point()`.

#### 5.636.4.2 template<typename \_CharT> virtual char\_type std::numpunct<\_CharT>::do\_decimal\_point( ) const [inline, protected, virtual]

Return decimal point character.

Returns a `char_type` to use as a decimal point. This function is a hook for derived classes to change the value returned.

#### Returns

*char\_type* representing a decimal point.

Definition at line 1795 of file `locale_facets.h`.

Referenced by `std::numpunct<_CharT>::decimal_point()`.

#### 5.636.4.3 template<typename \_CharT> virtual string\_type std::numpunct<\_CharT>::do\_falsename( ) const [inline, protected, virtual]

Return string representation of bool false.

Returns a `string_type` containing the text representation for false bool variables. This function is a hook for derived classes to change the value returned.

#### Returns

*string\_type* representing printed form of false.

Definition at line 1846 of file `locale_facets.h`.

Referenced by `std::numpunct<_CharT>::falsename()`.

#### 5.636.4.4 `template<typename _CharT> virtual string std::num_punct<_CharT>::do_grouping( ) const [inline, protected, virtual]`

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

##### See also

[grouping\(\)](#) for details.

##### Returns

String representing grouping specification.

Definition at line 1820 of file locale\_facets.h.

Referenced by `std::num_punct<_CharT>::grouping()`.

#### 5.636.4.5 `template<typename _CharT> virtual char_type std::num_punct<_CharT>::do_thousands_sep( ) const [inline, protected, virtual]`

Return thousands separator character.

Returns a `char_type` to use as a thousands separator. This function is a hook for derived classes to change the value returned.

##### Returns

*char\_type* representing a thousands separator.

Definition at line 1807 of file locale\_facets.h.

Referenced by `std::num_punct<_CharT>::thousands_sep()`.

#### 5.636.4.6 `template<typename _CharT> virtual string_type std::num_punct<_CharT>::do_truename( ) const [inline, protected, virtual]`

Return string representation of bool true.

Returns a `string_type` containing the text representation for true bool variables. This function is a hook for derived classes to change the value returned.

**Returns**

`string_type` representing printed form of true.

Definition at line 1833 of file `locale_facets.h`.

Referenced by `std::num_punct<_CharT>::truename()`.

**5.636.4.7 template<typename \_CharT> string\_type std::num\_punct<\_CharT>::falsename( ) const [inline]**

Return string representation of bool false.

This function returns a `string_type` containing the text representation for false bool variables. It does so by calling `num_punct<char_type>::do_falsename()`.

**Returns**

`string_type` representing printed form of false.

Definition at line 1778 of file `locale_facets.h`.

References `std::num_punct<_CharT>::do_falsename()`.

**5.636.4.8 template<typename \_CharT> string std::num\_punct<\_CharT>::grouping( ) const [inline]**

Return grouping specification.

This function returns a string representing groupings for the integer part of a number. Groupings indicate where thousands separators should be inserted in the integer part of a number.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the `grouping()` returns `"\003\002"` and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was `"32"`, this



would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling `num_punct<char_type>::do_grouping()`.

**Returns**

string representing grouping specification.

Definition at line 1752 of file locale\_facets.h.

References `std::num_punct<_CharT>::do_grouping()`.

**5.636.4.9 template<typename \_CharT> char\_type std::num\_punct<\_CharT>::thousands\_sep( ) const [inline]**

Return thousands separator character.

This function returns a `char_type` to use as a thousands separator. It does so by returning `num_punct<char_type>::do_thousands_sep()`.

**Returns**

`char_type` representing a thousands separator.

Definition at line 1721 of file locale\_facets.h.

References `std::num_punct<_CharT>::do_thousands_sep()`.

**5.636.4.10 template<typename \_CharT> string\_type std::num\_punct<\_CharT>::true\_name( ) const [inline]**

Return string representation of bool true.

This function returns a `string_type` containing the text representation for true bool variables. It does so by calling `num_punct<char_type>::do_true_name()`.

**Returns**

`string_type` representing printed form of true.

Definition at line 1765 of file locale\_facets.h.

References `std::num_punct<_CharT>::do_true_name()`.

### 5.636.5 Member Data Documentation

#### 5.636.5.1 template<typename \_CharT > locale::id std::numpunct< \_CharT >::id [static]

Numpunct facet id.

Definition at line 1658 of file locale\_facets.h.

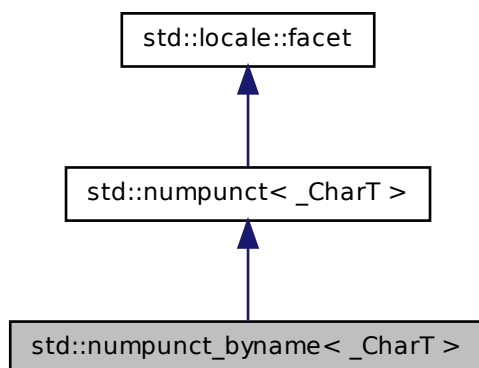
The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

### 5.637 std::numpunct\_byname< \_CharT > Class Template Reference

class [numpunct\\_byname](#) [22.2.3.2].

Inheritance diagram for std::numpunct\_byname< \_CharT >:



#### Public Types

- typedef \_\_numpunct\_cache< \_CharT > \_\_cache\_type
- typedef \_CharT [char\\_type](#)

- typedef `basic_string<_CharT>` `string_type`

### Public Member Functions

- `numpunct_byname` (const char \*\_\_s, size\_t \_\_refs=0)
- `char_type decimal_point` () const
- `string_type falsename` () const
- `string grouping` () const
- `char_type thousands_sep` () const
- `string_type truename` () const

### Static Public Attributes

- static `locale::id` `id`

### Protected Member Functions

- template<>  
void `_M_initialize_numpunct` (\_\_c\_locale \_\_cloc)
- template<>  
void `_M_initialize_numpunct` (\_\_c\_locale \_\_cloc)
- void `_M_initialize_numpunct` (\_\_c\_locale \_\_cloc=0)
- virtual `char_type do_decimal_point` () const
- virtual `string_type do_falsename` () const
- virtual `string do_grouping` () const
- virtual `char_type do_thousands_sep` () const
- virtual `string_type do_truename` () const

### Static Protected Member Functions

- static \_\_c\_locale `_S_clone_c_locale` (\_\_c\_locale &\_\_cloc) throw ()
- static void `_S_create_c_locale` (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void `_S_destroy_c_locale` (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale `_S_get_c_locale` ()
- static const char \* `_S_get_c_name` () throw ()
- static \_\_c\_locale `_S_lc_ctype_c_locale` (\_\_c\_locale \_\_cloc, const char \*\_\_s)

### Protected Attributes

- `__cache_type` \* `_M_data`

## Friends

- class `locale::_Impl`

### 5.637.1 Detailed Description

`template<typename _CharT> class std::num_punct_byname<_CharT>`

class `num_punct_byname` [22.2.3.2].

Definition at line 1875 of file `locale_facets.h`.

### 5.637.2 Member Typedef Documentation

**5.637.2.1** `template<typename _CharT> typedef _CharT  
std::num_punct_byname<_CharT>::char_type`

Public typedefs.

Reimplemented from `std::num_punct<_CharT>`.

Definition at line 1878 of file `locale_facets.h`.

**5.637.2.2** `template<typename _CharT> typedef basic_string<_CharT>  
std::num_punct_byname<_CharT>::string_type`

Public typedefs.

Reimplemented from `std::num_punct<_CharT>`.

Definition at line 1879 of file `locale_facets.h`.

### 5.637.3 Member Function Documentation

**5.637.3.1** `template<typename _CharT> char_type std::num_punct<_CharT>  
>::decimal_point( ) const [inline, inherited]`

Return decimal point character.

This function returns a `char_type` to use as a decimal point. It does so by returning  
returning `num_punct<char_type>::do_decimal_point()`.

**Returns**

*char\_type* representing a decimal point.

Definition at line 1708 of file `locale_facets.h`.

References `std::num_punct<_CharT>::do_decimal_point()`.

**5.637.3.2** `template<typename _CharT> virtual char_type std::num_punct<_CharT>::do_decimal_point( ) const [inline, protected, virtual, inherited]`

Return decimal point character.

Returns a *char\_type* to use as a decimal point. This function is a hook for derived classes to change the value returned.

**Returns**

*char\_type* representing a decimal point.

Definition at line 1795 of file `locale_facets.h`.

Referenced by `std::num_punct<_CharT>::decimal_point()`.

**5.637.3.3** `template<typename _CharT> virtual string_type std::num_punct<_CharT>::do_falsename( ) const [inline, protected, virtual, inherited]`

Return string representation of bool false.

Returns a *string\_type* containing the text representation for false bool variables. This function is a hook for derived classes to change the value returned.

**Returns**

*string\_type* representing printed form of false.

Definition at line 1846 of file `locale_facets.h`.

Referenced by `std::num_punct<_CharT>::falsename()`.

**5.637.3.4 template<typename \_CharT> virtual string std::num\_punct<\_CharT>::do\_grouping( ) const [inline, protected, virtual, inherited]**

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

**See also**

[grouping\(\)](#) for details.

**Returns**

String representing grouping specification.

Definition at line 1820 of file locale\_facets.h.

Referenced by std::num\_punct<\_CharT>::grouping().

**5.637.3.5 template<typename \_CharT> virtual char\_type std::num\_punct<\_CharT>::do\_thousands\_sep( ) const [inline, protected, virtual, inherited]**

Return thousands separator character.

Returns a char\_type to use as a thousands separator. This function is a hook for derived classes to change the value returned.

**Returns**

*char\_type* representing a thousands separator.

Definition at line 1807 of file locale\_facets.h.

Referenced by std::num\_punct<\_CharT>::thousands\_sep().

**5.637.3.6 template<typename \_CharT> virtual string\_type std::num\_punct<\_CharT>::do\_truename( ) const [inline, protected, virtual, inherited]**

Return string representation of bool true.

Returns a `string_type` containing the text representation for true bool variables. This function is a hook for derived classes to change the value returned.

**Returns**

`string_type` representing printed form of true.

Definition at line 1833 of file `locale_facets.h`.

Referenced by `std::num_punct<_CharT>::truename()`.

**5.637.3.7 `template<typename _CharT> string_type std::num_punct<_CharT>::falsename( ) const [inline, inherited]`**

Return string representation of bool false.

This function returns a `string_type` containing the text representation for false bool variables. It does so by calling `num_punct<char_type>::do_falsename()`.

**Returns**

`string_type` representing printed form of false.

Definition at line 1778 of file `locale_facets.h`.

References `std::num_punct<_CharT>::do_falsename()`.

**5.637.3.8 `template<typename _CharT> string std::num_punct<_CharT>::grouping( ) const [inline, inherited]`**

Return grouping specification.

This function returns a string representing groupings for the integer part of a number. Groupings indicate where thousands separators should be inserted in the integer part of a number.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the `grouping()` returns `"\003\002"` and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was `"32"`, this

would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling `num_punct<char_type>::do_grouping()`.

#### Returns

string representing grouping specification.

Definition at line 1752 of file locale\_facets.h.

References `std::num_punct<_CharT>::do_grouping()`.

#### 5.637.3.9 template<typename \_CharT> char\_type std::num\_punct<\_CharT>::thousands\_sep( ) const [inline, inherited]

Return thousands separator character.

This function returns a `char_type` to use as a thousands separator. It does so by returning `num_punct<char_type>::do_thousands_sep()`.

#### Returns

`char_type` representing a thousands separator.

Definition at line 1721 of file locale\_facets.h.

References `std::num_punct<_CharT>::do_thousands_sep()`.

#### 5.637.3.10 template<typename \_CharT> string\_type std::num\_punct<\_CharT>::true\_name( ) const [inline, inherited]

Return string representation of bool true.

This function returns a `string_type` containing the text representation for true bool variables. It does so by calling `num_punct<char_type>::do_true_name()`.

#### Returns

`string_type` representing printed form of true.

Definition at line 1765 of file locale\_facets.h.

References `std::num_punct<_CharT>::do_true_name()`.



### 5.637.4 Member Data Documentation

**5.637.4.1** `template<typename _CharT > locale::id std::numpunct< _CharT >::id [static, inherited]`

Numpunct facet id.

Definition at line 1658 of file locale\_facets.h.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

## 5.638 std::once\_flag Struct Reference

[once\\_flag](#)

### Public Member Functions

- **once\_flag** (const [once\\_flag](#) &)
- **once\_flag & operator=** (const [once\\_flag](#) &)

### Friends

- `template<typename _Callable , typename... _Args>`  
`void call\_once (once\_flag &__once, _Callable && __f, _Args &&... __args)`

### 5.638.1 Detailed Description

[once\\_flag](#)

Definition at line 755 of file mutex.

### 5.638.2 Friends And Related Function Documentation

**5.638.2.1** `template<typename _Callable , typename... _Args> void call\_once (`  
`once\_flag & __once, _Callable && __f, _Args &&... __args )`  
`[friend]`

[call\\_once](#)

Definition at line 797 of file `mutex`.

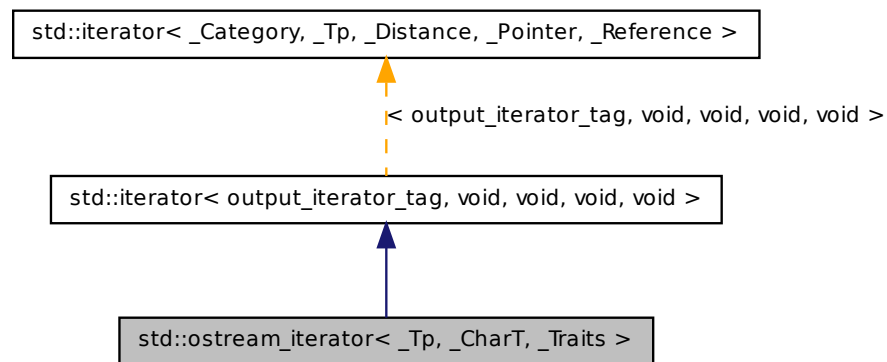
The documentation for this struct was generated from the following file:

- [mutex](#)

### 5.639 `std::ostream_iterator< _Tp, _CharT, _Traits >` Class Template Reference

Provides output iterator semantics for streams.

Inheritance diagram for `std::ostream_iterator< _Tp, _CharT, _Traits >`:



#### Public Types

- typedef void [difference\\_type](#)
- typedef [output\\_iterator\\_tag](#) [iterator\\_category](#)
- typedef void [pointer](#)
- typedef void [reference](#)
- typedef void [value\\_type](#)
  
- typedef [\\_CharT](#) [char\\_type](#)
- typedef [\\_Traits](#) [traits\\_type](#)
- typedef [basic\\_ostream< \\_CharT, \\_Traits >](#) [ostream\\_type](#)

## Public Member Functions

- `ostream_iterator` (`ostream_type` &\_\_s)
- `ostream_iterator` (`ostream_type` &\_\_s, const `_CharT` \*\_\_c)
- `ostream_iterator` (const `ostream_iterator` &\_\_obj)
- `ostream_iterator` & `operator*` ()
- `ostream_iterator` & `operator++` (int)
- `ostream_iterator` & `operator++` ()
- `ostream_iterator` & `operator=` (const `_Tp` &\_\_value)

### 5.639.1 Detailed Description

`template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>> class std::ostream_iterator< _Tp, _CharT, _Traits >`

Provides output iterator semantics for streams. This class provides an iterator to write to an ostream. The type `Tp` is the only type written by this iterator and there must be an `operator<<(Tp)` defined.

#### Parameters

***Tp*** The type to write to the ostream.

***CharT*** The ostream char\_type.

***Traits*** The ostream `char_traits`.

Definition at line 154 of file `stream_iterator.h`.

### 5.639.2 Member Typedef Documentation

**5.639.2.1** `template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>> typedef _CharT std::ostream_iterator< _Tp, _CharT, _Traits >::char_type`

Public typedef.

Definition at line 160 of file `stream_iterator.h`.

**5.639.2.2** `typedef void std::iterator< output_iterator_tag , void , void , void , void >::difference_type [inherited]`

Distance between iterators is represented as this type.

Definition at line 126 of file stl\_iterator\_base\_types.h.

**5.639.2.3** `typedef output_iterator_tag std::iterator< output_iterator_tag , void , void , void , void >::iterator_category [inherited]`

One of the [tag types](#).

Definition at line 122 of file stl\_iterator\_base\_types.h.

**5.639.2.4** `template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>> typedef basic_ostream<_CharT, _Traits> std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_type`

Public typedef.

Definition at line 162 of file stream\_iterator.h.

**5.639.2.5** `typedef void std::iterator< output_iterator_tag , void , void , void , void >::pointer [inherited]`

This type represents a pointer-to-value\_type.

Definition at line 128 of file stl\_iterator\_base\_types.h.

**5.639.2.6** `typedef void std::iterator< output_iterator_tag , void , void , void , void >::reference [inherited]`

This type represents a reference-to-value\_type.

Definition at line 130 of file stl\_iterator\_base\_types.h.

**5.639.2.7** `template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>> typedef _Traits std::ostream_iterator< _Tp, _CharT, _Traits >::traits_type`

Public typedef.

Definition at line 161 of file stream\_iterator.h.

**5.639.2.8** `typedef void std::iterator< output_iterator_tag , void , void , void , void >::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 124 of file stl\_iterator\_base\_types.h.

### **5.639.3 Constructor & Destructor Documentation**

**5.639.3.1** `template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>> std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator ( ostream_type & __s ) [inline]`

Construct from an ostream.

Definition at line 171 of file stream\_iterator.h.

**5.639.3.2** `template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>> std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator ( ostream_type & __s, const _CharT * __c ) [inline]`

Construct from an ostream.

The delimiter string *c* is written to the stream after every *Tp* written to the stream. The delimiter is not copied, and thus must not be destroyed while this iterator is in use.

#### **Parameters**

*s* Underlying ostream to write to.

*c* CharT delimiter string to insert.

Definition at line 183 of file stream\_iterator.h.

## 5.640 `std::ostreambuf_iterator< _CharT, _Traits >` Class Template Reference

**5.639.3.3** `template<typename _Tp , typename _CharT = char, typename  
_Traits = char_traits<_CharT>> std::ostream_iterator< _Tp,  
_CharT, _Traits >::ostream_iterator ( const ostream_iterator< _Tp,  
_CharT, _Traits > & __obj ) [inline]`

Copy constructor.

Definition at line 187 of file `stream_iterator.h`.

### 5.639.4 Member Function Documentation

**5.639.4.1** `template<typename _Tp , typename _CharT = char, typename  
_Traits = char_traits<_CharT>> ostream_iterator&  
std::ostream_iterator< _Tp, _CharT, _Traits >::operator= ( const  
_Tp & __value ) [inline]`

Writes *value* to underlying ostream using operator<<. If /// constructed with delimiter string, writes delimiter to ostream.

Definition at line 193 of file `stream_iterator.h`.

The documentation for this class was generated from the following file:

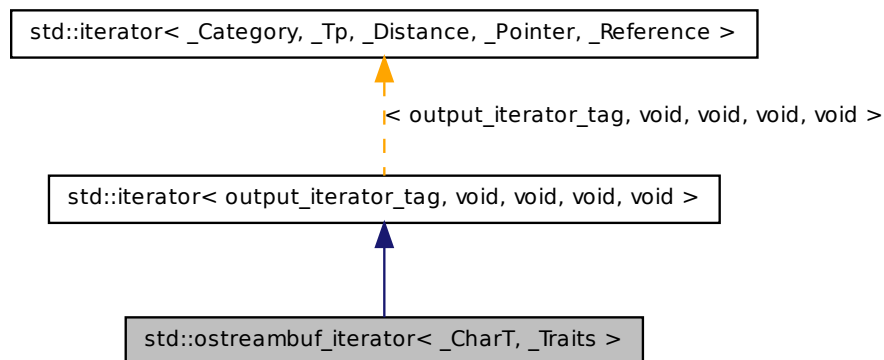
- [stream\\_iterator.h](#)

## 5.640 `std::ostreambuf_iterator< _CharT, _Traits >` Class Template Reference

Provides output iterator semantics for streambufs.

## 5.640 std::ostreambuf\_iterator< \_CharT, \_Traits > Class Template Reference 132

Inheritance diagram for std::ostreambuf\_iterator< \_CharT, \_Traits >:



### Public Types

- typedef void [difference\\_type](#)
- typedef [output\\_iterator\\_tag](#) [iterator\\_category](#)
- typedef void [pointer](#)
- typedef void [reference](#)
- typedef void [value\\_type](#)
  
- typedef [\\_CharT](#) [char\\_type](#)
- typedef [\\_Traits](#) [traits\\_type](#)
- typedef [basic\\_streambuf< \\_CharT, \\_Traits >](#) [streambuf\\_type](#)
- typedef [basic\\_ostream< \\_CharT, \\_Traits >](#) [ostream\\_type](#)

### Public Member Functions

- [ostreambuf\\_iterator](#) ([ostream\\_type](#) &\_\_s) throw ()
- [ostreambuf\\_iterator](#) ([streambuf\\_type](#) \*\_\_s) throw ()
- [ostreambuf\\_iterator](#) & [M\\_put](#) (const [\\_CharT](#) \*\_\_ws, [streamsize](#) \_\_len)
- bool [failed](#) () const throw ()
- [ostreambuf\\_iterator](#) & [operator\\*](#) ()
- [ostreambuf\\_iterator](#) & [operator++](#) ()
- [ostreambuf\\_iterator](#) & [operator++](#) (int)
- [ostreambuf\\_iterator](#) & [operator=](#) ([\\_CharT](#) \_\_c)

## 5.640 `std::ostreambuf_iterator<_CharT, _Traits>` Class Template Reference

### Friends

- `template<typename _CharT2 >`  
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, ostreambuf_iterator< _CharT2 >::__type >::copy (istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, ostreambuf_iterator< _CharT2 >)`

### 5.640.1 Detailed Description

`template<typename _CharT, typename _Traits> class std::ostreambuf_iterator< _CharT, _Traits >`

Provides output iterator semantics for streambufs.

Definition at line 206 of file `streambuf_iterator.h`.

### 5.640.2 Member Typedef Documentation

**5.640.2.1** `template<typename _CharT, typename _Traits > typedef _CharT std::ostreambuf_iterator< _CharT, _Traits >::char_type`

Public typedefs.

Definition at line 213 of file `streambuf_iterator.h`.

**5.640.2.2** `typedef void std::iterator< output_iterator_tag, void, void, void, void >::difference_type [inherited]`

Distance between iterators is represented as this type.

Definition at line 126 of file `stl_iterator_base_types.h`.

**5.640.2.3** `typedef output_iterator_tag std::iterator< output_iterator_tag, void, void, void, void >::iterator_category [inherited]`

One of the [tag types](#).

Definition at line 122 of file `stl_iterator_base_types.h`.



## **5.640 std::ostreambuf\_iterator< \_CharT, \_Traits > Class Template Reference**

**5.640.2.4** `template<typename _CharT , typename _Traits > typedef  
basic_ostream<_CharT, _Traits> std::ostreambuf_iterator<  
_CharT, _Traits >::ostream_type`

Public typedefs.

Definition at line 216 of file streambuf\_iterator.h.

**5.640.2.5** `typedef void std::iterator< output_iterator_tag , void , void , void ,  
void >::pointer [inherited]`

This type represents a pointer-to-value\_type.

Definition at line 128 of file stl\_iterator\_base\_types.h.

**5.640.2.6** `typedef void std::iterator< output_iterator_tag , void , void , void ,  
void >::reference [inherited]`

This type represents a reference-to-value\_type.

Definition at line 130 of file stl\_iterator\_base\_types.h.

**5.640.2.7** `template<typename _CharT , typename _Traits > typedef  
basic_streambuf<_CharT, _Traits> std::ostreambuf_iterator<  
_CharT, _Traits >::streambuf_type`

Public typedefs.

Definition at line 215 of file streambuf\_iterator.h.

**5.640.2.8** `template<typename _CharT , typename _Traits > typedef _Traits  
std::ostreambuf_iterator< _CharT, _Traits >::traits_type`

Public typedefs.

Definition at line 214 of file streambuf\_iterator.h.

## 5.640 `std::ostreambuf_iterator<_CharT, _Traits>` Class Template Reference

**5.640.2.9** `typedef void std::iterator< output_iterator_tag , void , void , void , void >::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 124 of file `stl_iterator_base_types.h`.

### 5.640.3 Constructor & Destructor Documentation

**5.640.3.1** `template<typename _CharT , typename _Traits >  
std::ostreambuf_iterator< _CharT, _Traits >::ostreambuf_iterator (   
ostream_type & __s ) throw () [inline]`

Construct output iterator from ostream.

Definition at line 231 of file `streambuf_iterator.h`.

**5.640.3.2** `template<typename _CharT , typename _Traits >  
std::ostreambuf_iterator< _CharT, _Traits >::ostreambuf_iterator (   
streambuf_type * __s ) throw () [inline]`

Construct output iterator from streambuf.

Definition at line 235 of file `streambuf_iterator.h`.

### 5.640.4 Member Function Documentation

**5.640.4.1** `template<typename _CharT , typename _Traits > bool  
std::ostreambuf_iterator< _CharT, _Traits >::failed ( ) const throw  
() [inline]`

Return true if previous `operator=()` failed.

Definition at line 265 of file `streambuf_iterator.h`.

## 5.640 `std::ostreambuf_iterator< _CharT, _Traits >` Class Template Reference

**5.640.4.2** `template<typename _CharT , typename _Traits >  
ostreambuf_iterator& std::ostreambuf_iterator< _CharT, _Traits  
>::operator* ( ) [inline]`

Return \*this.

Definition at line 250 of file `streambuf_iterator.h`.

**5.640.4.3** `template<typename _CharT , typename _Traits >  
ostreambuf_iterator& std::ostreambuf_iterator< _CharT, _Traits  
>::operator++ ( ) [inline]`

Return \*this.

Definition at line 260 of file `streambuf_iterator.h`.

**5.640.4.4** `template<typename _CharT , typename _Traits >  
ostreambuf_iterator& std::ostreambuf_iterator< _CharT, _Traits  
>::operator++ ( int ) [inline]`

Return \*this.

Definition at line 255 of file `streambuf_iterator.h`.

**5.640.4.5** `template<typename _CharT , typename _Traits >  
ostreambuf_iterator& std::ostreambuf_iterator< _CharT, _Traits  
>::operator= ( _CharT __c ) [inline]`

Write character to streambuf. Calls `streambuf.sputc()`.

Definition at line 240 of file `streambuf_iterator.h`.

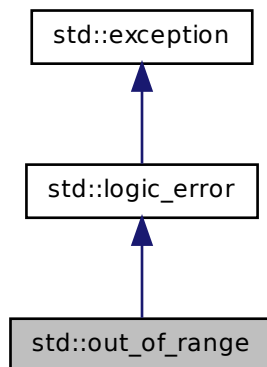
References `std::basic_streambuf< _CharT, _Traits >::sputc()`.

The documentation for this class was generated from the following file:

- [streambuf\\_iterator.h](#)

## 5.641 `std::out_of_range` Class Reference

Inheritance diagram for `std::out_of_range`:



### Public Member Functions

- **`out_of_range`** (const [string](#) &\_\_arg)
- virtual const char \* **`what`** () const throw ()

#### 5.641.1 Detailed Description

This represents an argument whose value is not within the expected range (e.g., boundary checks in [basic\\_string](#)).

Definition at line 99 of file `stdexcept`.

#### 5.641.2 Member Function Documentation

##### 5.641.2.1 virtual const char\* `std::logic_error::what` ( ) const throw () [virtual, inherited]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future\\_error](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

## 5.642 `std::output_iterator_tag` Struct Reference

Marking output iterators.

### 5.642.1 Detailed Description

Marking output iterators.

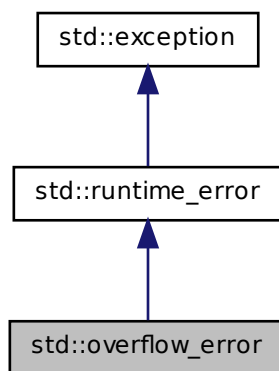
Definition at line 93 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 5.643 `std::overflow_error` Class Reference

Inheritance diagram for `std::overflow_error`:



## 5.644 `std::owner_less< shared_ptr< _Tp > >` Struct Template Reference 3139

### Public Member Functions

- **overflow\_error** (const [string](#) &\_\_arg)
- virtual const char \* **what** () const throw ()

#### 5.643.1 Detailed Description

Thrown to indicate arithmetic overflow.

Definition at line 136 of file `stdexcept`.

#### 5.643.2 Member Function Documentation

##### 5.643.2.1 `virtual const char* std::runtime_error::what ( ) const throw ()` [[virtual](#), [inherited](#)]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

## 5.644 `std::owner_less< shared_ptr< _Tp > >` Struct Template Reference

Partial specialization of `owner_less` for [shared\\_ptr](#).

Inherits `std::_Sp_owner_less< shared_ptr< _Tp >, weak_ptr< _Tp > >`.

### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef bool [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

### Public Member Functions

- bool **operator**() (const `_Tp` &\_\_lhs, const `_Tp` &\_\_rhs) const
- bool **operator**() (const `_Tp1` &\_\_lhs, const `_Tp` &\_\_rhs) const
- bool **operator**() (const `_Tp` &\_\_lhs, const `_Tp1` &\_\_rhs) const

### **5.644.1 Detailed Description**

**template<typename \_Tp> struct std::owner\_less< shared\_ptr< \_Tp > >**

Partial specialization of owner\_less for [shared\\_ptr](#).

Definition at line 460 of file shared\_ptr.h.

### **5.644.2 Member Typedef Documentation**

**5.644.2.1 typedef \_Tp std::binary\_function< \_Tp , \_Tp , bool  
>::first\_argument\_type [inherited]**

first\_argument\_type is the type of the first argument

Definition at line 118 of file stl\_function.h.

**5.644.2.2 typedef bool std::binary\_function< \_Tp , \_Tp , bool >::result\_type  
[inherited]**

result\_type is the return type

Definition at line 124 of file stl\_function.h.

**5.644.2.3 typedef \_Tp std::binary\_function< \_Tp , \_Tp , bool  
>::second\_argument\_type [inherited]**

second\_argument\_type is the type of the second argument

Definition at line 121 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [shared\\_ptr.h](#)

## **5.645 std::owner\_less< weak\_ptr< \_Tp > > Struct Template Reference**

Partial specialization of owner\_less for [weak\\_ptr](#).

Inherits std::\_Sp\_owner\_less< weak\_ptr< \_Tp >, shared\_ptr< \_Tp > >.

### Public Types

- typedef `_Tp` `first_argument_type`
- typedef `bool` `result_type`
- typedef `_Tp` `second_argument_type`

### Public Member Functions

- `bool operator()` (`const _Tp &__lhs, const _Tp &__rhs`) `const`
- `bool operator()` (`const _Tp1 &__lhs, const _Tp &__rhs`) `const`
- `bool operator()` (`const _Tp &__lhs, const _Tp1 &__rhs`) `const`

#### 5.645.1 Detailed Description

`template<typename _Tp> struct std::owner_less< weak_ptr< _Tp > >`

Partial specialization of `owner_less` for `weak_ptr`.

Definition at line 466 of file `shared_ptr.h`.

#### 5.645.2 Member Typedef Documentation

**5.645.2.1** `typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type [inherited]`

`first_argument_type` is the type of the first argument

Definition at line 118 of file `stl_function.h`.

**5.645.2.2** `typedef bool std::binary_function< _Tp , _Tp , bool >::result_type [inherited]`

`result_type` is the return type

Definition at line 124 of file `stl_function.h`.

**5.645.2.3** `typedef _Tp std::binary_function< _Tp , _Tp , bool >::second_argument_type [inherited]`



## 5.646 `std::packaged_task< _Res(_ArgTypes...)>` Class Template Reference 3142

`second_argument_type` is the type of the second argument

Definition at line 121 of file `std_function.h`.

The documentation for this struct was generated from the following file:

- [shared\\_ptr.h](#)

## 5.646 `std::packaged_task< _Res(_ArgTypes...)>` Class Template Reference

`packaged_task`

### Public Types

- `typedef _Res result_type`

### Public Member Functions

- `template<typename _Fn >`  
`packaged_task` (`const _Fn &__fn`)
- `packaged_task` (`_Res(*__fn)(_ArgTypes...)`)
- `packaged_task` (`packaged_task &&__other`)
- `template<typename _Fn, typename _Allocator >`  
`packaged_task` (`allocator_arg_t __tag, const _Allocator &__a, _Fn __fn`)
- `template<typename _Fn >`  
`packaged_task` (`_Fn &&__fn`)
- `packaged_task` (`packaged_task &`)
- `future< _Res >` `get_future` ()
- `operator bool` () `const`
- `void operator()` (`_ArgTypes... __args`)
- `packaged_task &` `operator=` (`packaged_task &`)
- `packaged_task &` `operator=` (`packaged_task &&__other`)
- `void reset` ()
- `void swap` (`packaged_task &__other`)

### 5.646.1 Detailed Description

`template<typename _Res, typename... _ArgTypes> class std::packaged_task< _Res(_ArgTypes...)>`

`packaged_task`

Definition at line 1195 of file future.

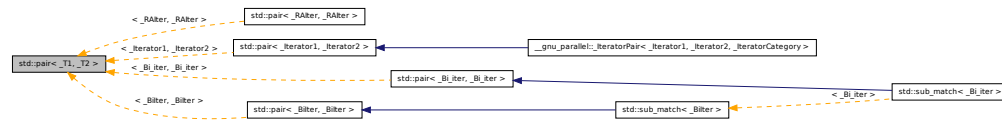
The documentation for this class was generated from the following file:

- [future](#)

## 5.647 std::pair< \_T1, \_T2 > Struct Template Reference

Struct holding two objects of arbitrary type.

Inheritance diagram for std::pair< \_T1, \_T2 >:



### Public Types

- typedef `_T1` **first\_type**
- typedef `_T2` **second\_type**

### Public Member Functions

- constexpr `pair` ()
- constexpr `pair` (const `_T1` &\_\_a, const `_T2` &\_\_b)
- constexpr `pair` (const `pair` &)
- template<class `_U1`, class `_U2` >  
`pair` (`pair`< `_U1`, `_U2` > &&\_\_p)
- template<class... `_Args1`, class... `_Args2` >  
`pair` (`piecewise_construct_t`, `tuple`< `_Args1...` > \_\_first, `tuple`< `_Args2...` > \_\_second)
- template<class `_U1`, class = typename `std::enable_if`<`std::is_convertible`< `_U1`, `_T1` >::value>::type>  
`pair` (`_U1` &&\_\_x, const `_T2` &\_\_y)
- template<class `_U1`, class `_U2` >  
constexpr `pair` (const `pair`< `_U1`, `_U2` > &&\_\_p)
- template<class `_U2`, class = typename `std::enable_if`<`std::is_convertible`< `_U2`, `_T2` >::value>::type>  
`pair` (const `_T1` &\_\_x, `_U2` &&\_\_y)

- `template<class _U1 , class _U2 , class = typename std::enable_if<std::is_convertible<_U1, _T1>::value && std::is_convertible<_U2, _T2>::value>::type>`  
`pair ( _U1 &&__x, _U2 &&__y)`
- `pair & operator= (const pair &__p)`
- `pair & operator= (pair &&__p)`
- `template<class _U1 , class _U2 >`  
`pair & operator= (const pair< _U1, _U2 > &__p)`
- `template<class _U1 , class _U2 >`  
`pair & operator= (pair< _U1, _U2 > &&__p)`
- `void swap (pair &__p)`

### Public Attributes

- `_T1` `first`
- `_T2` `second`

#### 5.647.1 Detailed Description

`template<class _T1, class _T2> struct std::pair< _T1, _T2 >`

Struct holding two objects of arbitrary type.

Definition at line 87 of file `stl_pair.h`.

#### 5.647.2 Member Typedef Documentation

**5.647.2.1** `template<class _T1, class _T2> typedef _T2 std::pair< _T1, _T2 >::second_type`

`first_type` is the first bound type

Definition at line 90 of file `stl_pair.h`.

#### 5.647.3 Constructor & Destructor Documentation

**5.647.3.1** `template<class _T1, class _T2> constexpr std::pair< _T1, _T2 >::pair ( ) [inline]`

`second` is a copy of the second object

The default constructor creates `first` and `second` using their respective default constructors.

Definition at line 99 of file `stl_pair.h`.

**5.647.3.2** `template<class _T1, class _T2> constexpr std::pair<_T1, _T2>::pair ( const _T1 & __a, const _T2 & __b ) [inline]`

Two objects may be passed to a `pair` constructor to be copied.

Definition at line 103 of file `stl_pair.h`.

**5.647.3.3** `template<class _T1, class _T2> template<class _U1, class _U2 > constexpr std::pair<_T1, _T2>::pair ( const pair<_U1, _U2> & __p ) [inline]`

There is also a templated copy ctor for the `pair` class itself.

Definition at line 108 of file `stl_pair.h`.

#### 5.647.4 Member Data Documentation

**5.647.4.1** `template<class _T1, class _T2> _T1 std::pair<_T1, _T2>::first`

`second_type` is the second bound type

Definition at line 92 of file `stl_pair.h`.

Referenced by `std::_Temporary_buffer<_ForwardIterator, _Tp>::_Temporary_buffer()`, `std::set<_StateIdT>::insert()`, and `std::operator==()`.

**5.647.4.2** `template<class _T1, class _T2> _T2 std::pair<_T1, _T2>::second`

`first` is a copy of the first object

Definition at line 93 of file `stl_pair.h`.

Referenced by `std::_Temporary_buffer<_ForwardIterator, _Tp>::_Temporary_buffer()`, `std::set<_StateIdT>::insert()`, and `std::operator==()`.

The documentation for this struct was generated from the following files:

- [stl\\_pair.h](#)
- [tuple](#)

## 5.648 `std::piecewise_constant_distribution<_RealType>` Class Template Reference

A [piecewise\\_constant\\_distribution](#) random number distribution.

### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_RealType` [result\\_type](#)

### Public Member Functions

- `template<typename _InputIteratorB, typename _InputIteratorW >`  
**`piecewise_constant_distribution`** (`_InputIteratorB __bfirst, _InputIteratorB __bend, _InputIteratorW __wbegin`)
- `template<typename _Func >`  
**`piecewise_constant_distribution`** (`size_t __nw, _RealType __xmin, _RealType __xmax, _Func __fw`)
- **`piecewise_constant_distribution`** (`const param\_type &__p`)
- `template<typename _Func >`  
**`piecewise_constant_distribution`** (`initializer_list<_RealType> __bl, _Func __fw`)
- `std::vector<double> densities () const`
- `std::vector<_RealType> intervals () const`
- `result\_type max () const`
- `result\_type min () const`
- `template<typename _UniformRandomNumberGenerator >`  
`result\_type operator() (_UniformRandomNumberGenerator &__urng, const param\_type &__p)`
- `template<typename _UniformRandomNumberGenerator >`  
`result\_type operator() (_UniformRandomNumberGenerator &__urng)`
- `param\_type param () const`
- `void param (const param\_type &__param)`
- `void reset ()`

### Friends

- `template<typename _RealType1, typename _CharT, typename _Traits >`  
`std::basic\_ostream<_CharT, _Traits> & operator<< (std::basic\_ostream<_CharT, _Traits> &, const std::piecewise\_constant\_distribution<_RealType1> &)`

- `template<typename _RealType1, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _  
CharT, _Traits > &, std::piecewise_constant_distribution< _RealType1 > &)`

### 5.648.1 Detailed Description

`template<typename _RealType = double> class std::piecewise_constant_distribution< _RealType >`

A [piecewise\\_constant\\_distribution](#) random number distribution. The formula for the piecewise constant probability mass function is

Definition at line 4861 of file random.h.

### 5.648.2 Member Typedef Documentation

**5.648.2.1** `template<typename _RealType = double> typedef _RealType  
std::piecewise_constant_distribution< _RealType >::result_type`

The type of the range of the distribution.

Definition at line 4868 of file random.h.

### 5.648.3 Member Function Documentation

**5.648.3.1** `template<typename _RealType = double> std::vector<double>  
std::piecewise_constant_distribution< _RealType >::densities ( )  
const [inline]`

Returns a vector of the probability densities.

Definition at line 4982 of file random.h.

References `std::vector<_Tp, _Alloc>::empty()`.

**5.648.3.2** `template<typename _RealType = double> std::vector<_RealType>  
std::piecewise_constant_distribution< _RealType >::intervals ( )  
const [inline]`

Returns a vector of the intervals.

Definition at line 4966 of file random.h.

References `std::vector<_Tp, _Alloc>::empty()`.

**5.648.3.3** `template<typename _RealType = double> result_type  
std::piecewise_constant_distribution<_RealType>::max ( ) const  
[inline]`

Returns the least upper bound value of the distribution.

Definition at line 5017 of file `random.h`.

References `std::vector<_Tp, _Alloc>::back()`, and `std::vector<_Tp, _Alloc>::empty()`.

**5.648.3.4** `template<typename _RealType = double> result_type  
std::piecewise_constant_distribution<_RealType>::min ( ) const  
[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 5007 of file `random.h`.

References `std::vector<_Tp, _Alloc>::empty()`, and `std::vector<_Tp, _Alloc>::front()`.

**5.648.3.5** `template<typename _RealType = double> template<typename  
_UniformRandomNumberGenerator> result_type  
std::piecewise_constant_distribution<_RealType>::operator() (   
_UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 5028 of file `random.h`.

References `std::piecewise_constant_distribution<_RealType>::operator()()`, and `std::piecewise_constant_distribution<_RealType>::param()`.

Referenced by `std::piecewise_constant_distribution<_RealType>::operator()()`.

**5.648.3.6** `template<typename _RealType = double> void  
std::piecewise_constant_distribution<_RealType>::param ( const  
param_type & __param ) [inline]`

Sets the parameter set of the distribution.

#### Parameters

`__param` The new parameter set of the distribution.

Definition at line 5000 of file random.h.

**5.648.3.7** `template<typename _RealType = double> param_type  
std::piecewise_constant_distribution<_RealType>::param ( )  
const [inline]`

Returns the parameter set of the distribution.

Definition at line 4992 of file random.h.

Referenced by `std::piecewise_constant_distribution<_RealType>::operator()()`, and `std::operator==( )`.

**5.648.3.8** `template<typename _RealType = double> void  
std::piecewise_constant_distribution<_RealType>::reset ( )  
[inline]`

Resets the distribution state.

Definition at line 4959 of file random.h.

#### 5.648.4 Friends And Related Function Documentation

**5.648.4.1** `template<typename _RealType = double> template<typename  
_RealType1 , typename _CharT , typename _Traits >  
std::basic_ostream<_CharT, _Traits>& operator<<  
( std::basic_ostream<_CharT, _Traits> & , const  
std::piecewise_constant_distribution<_RealType1> & )  
[friend]`



## 5.649 `std::piecewise_constant_distribution<_RealType>::param_type` Struct Reference 3150

---

Inserts a `piecewise_constant_distribution` random number distribution `__x` into the output stream `__os`.

### Parameters

- `__os` An output stream.
- `__x` A `piecewise_constant_distribution` random number distribution.

### Returns

The output stream with the state of `__x` inserted or in an error state.

**5.648.4.2** `template<typename _RealType = double> template<typename _RealType1, typename _CharT, typename _Traits> std::basic_istream<_CharT, _Traits>& operator>> ( std::basic_istream<_CharT, _Traits> &, std::piecewise_constant_distribution<_RealType1> & ) [friend]`

Extracts a `piecewise_constant_distribution` random number distribution `__x` from the input stream `__is`.

### Parameters

- `__is` An input stream.
- `__x` A `piecewise_constant_distribution` random number generator engine.

### Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.649 `std::piecewise_constant_distribution<_RealType>::param_type` Struct Reference

### Public Types

- typedef [piecewise\\_constant\\_distribution](#)<\_RealType> **distribution\_type**

## Public Member Functions

- `template<typename _InputIteratorB, typename _InputIteratorW >`  
**param\_type** (`_InputIteratorB __bfirst, _InputIteratorB __bend, _InputIteratorW __wbegin`)
- `template<typename _Func >`  
**param\_type** (`size_t __nw, _RealType __xmin, _RealType __xmax, _Func __fw`)
- **param\_type** (`const param_type &`)
- `template<typename _Func >`  
**param\_type** (`initializer_list< _RealType > __bi, _Func __fw`)
- `std::vector< double > densities` () const
- `std::vector< _RealType > intervals` () const
- `param_type & operator=` (`const param_type &`)

## Friends

- `bool operator==` (`const param_type &__p1, const param_type &__p2`)
- `class piecewise_constant_distribution< _RealType >`

### 5.649.1 Detailed Description

`template<typename _RealType = double> struct std::piecewise_constant_distribution< _RealType >::param_type`

Parameter type.

Definition at line 4870 of file `random.h`.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.650 `std::piecewise_construct_t` Struct Reference

[piecewise\\_construct\\_t](#)

### 5.650.1 Detailed Description

[piecewise\\_construct\\_t](#)

Definition at line 72 of file `stl_pair.h`.

The documentation for this struct was generated from the following file:

- [stl\\_pair.h](#)

## 5.651 `std::piecewise_linear_distribution<_RealType>` Class Template Reference

A [piecewise\\_linear\\_distribution](#) random number distribution.

### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_RealType` [result\\_type](#)

### Public Member Functions

- `template<typename _InputIteratorB, typename _InputIteratorW >`  
**`piecewise_linear_distribution`** (`_InputIteratorB __bfirst`, `_InputIteratorB __bend`, `_InputIteratorW __wbegin`)
- `template<typename _Func >`  
**`piecewise_linear_distribution`** (`size_t __nw`, `_RealType __xmin`, `_RealType __xmax`, `_Func __fw`)
- **`piecewise_linear_distribution`** (`const` [param\\_type](#) &\_\_p)
- `template<typename _Func >`  
**`piecewise_linear_distribution`** ([initializer\\_list](#)< `_RealType` > \_\_bl, `_Func __fw`)
- [std::vector](#)< `double` > [densities](#) () `const`
- [std::vector](#)< `_RealType` > [intervals](#) () `const`
- [result\\_type](#) [max](#) () `const`
- [result\\_type](#) [min](#) () `const`
- `template<typename _UniformRandomNumberGenerator >`  
[result\\_type](#) **`operator()`** (`_UniformRandomNumberGenerator &__urng`, `const` [param\\_type](#) &\_\_p)
- `template<typename _UniformRandomNumberGenerator >`  
[result\\_type](#) **`operator()`** (`_UniformRandomNumberGenerator &__urng`)
- [param\\_type](#) [param](#) () `const`
- `void` [param](#) (`const` [param\\_type](#) &\_\_param)
- `void` [reset](#) ()

## Friends

- `template<typename _RealType1, typename _CharT, typename _Traits >`  
`std::basic_ostream<_CharT, _Traits> & operator<< (std::basic_ostream<_`  
`CharT, _Traits> &, const std::piecewise_linear_distribution<_RealType1 >`  
`&)`
- `template<typename _RealType1, typename _CharT, typename _Traits >`  
`std::basic_istream<_CharT, _Traits> & operator>> (std::basic_istream<_`  
`CharT, _Traits> &, std::piecewise_linear_distribution<_RealType1 > &)`

### 5.651.1 Detailed Description

`template<typename _RealType = double> class std::piecewise_linear_distribution<_RealType>`

A [piecewise\\_linear\\_distribution](#) random number distribution. The formula for the piecewise linear probability mass function is

Definition at line 5100 of file `random.h`.

### 5.651.2 Member Typedef Documentation

**5.651.2.1** `template<typename _RealType = double> typedef _RealType`  
`std::piecewise_linear_distribution<_RealType>::result_type`

The type of the range of the distribution.

Definition at line 5107 of file `random.h`.

### 5.651.3 Member Function Documentation

**5.651.3.1** `template<typename _RealType = double> std::vector<double>`  
`std::piecewise_linear_distribution<_RealType>::densities ( ) const`  
`[inline]`

Return a vector of the probability densities of the distribution.

Definition at line 5224 of file `random.h`.

References `std::vector<_Tp, _Alloc>::empty()`.

**5.651.3.2** `template<typename _RealType = double> std::vector<_RealType>  
std::piecewise_linear_distribution<_RealType>::intervals ( ) const  
[inline]`

Return the intervals of the distribution.

Definition at line 5207 of file random.h.

References `std::vector<_Tp, _Alloc>::empty()`.

**5.651.3.3** `template<typename _RealType = double> result_type  
std::piecewise_linear_distribution<_RealType>::max ( ) const  
[inline]`

Returns the least upper bound value of the distribution.

Definition at line 5259 of file random.h.

References `std::vector<_Tp, _Alloc>::back()`, and `std::vector<_Tp, _Alloc>::empty()`.

**5.651.3.4** `template<typename _RealType = double> result_type  
std::piecewise_linear_distribution<_RealType>::min ( ) const  
[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 5249 of file random.h.

References `std::vector<_Tp, _Alloc>::empty()`, and `std::vector<_Tp, _Alloc>::front()`.

**5.651.3.5** `template<typename _RealType = double> template<typename  
_UniformRandomNumberGenerator> result_type  
std::piecewise_linear_distribution<_RealType>::operator() (   
_UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 5270 of file random.h.

References `std::piecewise_linear_distribution<_RealType>::operator()`, and `std::piecewise_linear_distribution<_RealType>::param()`.

Referenced by `std::piecewise_linear_distribution<_RealType>::operator()`.

**5.651.3.6** `template<typename _RealType = double> void  
std::piecewise_linear_distribution<_RealType>::param ( const  
param_type & __param ) [inline]`

Sets the parameter set of the distribution.

#### Parameters

`__param` The new parameter set of the distribution.

Definition at line 5242 of file `random.h`.

**5.651.3.7** `template<typename _RealType = double> param_type  
std::piecewise_linear_distribution<_RealType>::param ( ) const  
[inline]`

Returns the parameter set of the distribution.

Definition at line 5234 of file `random.h`.

Referenced by `std::piecewise_linear_distribution<_RealType>::operator()`, and `std::operator==( )`.

**5.651.3.8** `template<typename _RealType = double> void  
std::piecewise_linear_distribution<_RealType>::reset ( )  
[inline]`

Resets the distribution state.

Definition at line 5200 of file `random.h`.

#### 5.651.4 Friends And Related Function Documentation

**5.651.4.1** `template<typename _RealType = double> template<typename  
_RealType1 , typename _CharT , typename _Traits >  
std::basic_ostream<_CharT, _Traits>& operator<<  
( std::basic_ostream< _CharT, _Traits > & , const  
std::piecewise_linear_distribution< _RealType1 > & ) [friend]`

Inserts a `piecewise_linear_distribution` random number distribution `__x` into the output stream `__os`.

##### Parameters

`__os` An output stream.  
`__x` A `piecewise_linear_distribution` random number distribution.

##### Returns

The output stream with the state of `__x` inserted or in an error state.

**5.651.4.2** `template<typename _RealType = double> template<typename  
_RealType1 , typename _CharT , typename _Traits >  
std::basic_istream<_CharT, _Traits>& operator>>  
( std::basic_istream< _CharT, _Traits > & ,  
std::piecewise_linear_distribution< _RealType1 > & ) [friend]`

Extracts a `piecewise_linear_distribution` random number distribution `__x` from the input stream `__is`.

##### Parameters

`__is` An input stream.  
`__x` A `piecewise_linear_distribution` random number generator engine.

##### Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.652 `std::piecewise_linear_distribution<_RealType>::param_type` Struct Reference

### Public Types

- typedef [piecewise\\_linear\\_distribution](#)<\_RealType> **distribution\_type**

### Public Member Functions

- template<typename \_InputIteratorB, typename \_InputIteratorW >  
**param\_type** (\_InputIteratorB \_\_bfirst, \_InputIteratorB \_\_bend, \_InputIteratorW \_\_wbegin)
- template<typename \_Func >  
**param\_type** (size\_t \_\_nw, \_RealType \_\_xmin, \_RealType \_\_xmax, \_Func \_\_fw)
- param\_type** (const [param\\_type](#) &)
- template<typename \_Func >  
**param\_type** ([initializer\\_list](#)<\_RealType> \_\_bl, \_Func \_\_fw)
- [std::vector](#)<double> **densities** () const
- [std::vector](#)<\_RealType> **intervals** () const
- [param\\_type](#) & **operator=** (const [param\\_type](#) &)

### Friends

- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- class **piecewise\_linear\_distribution**<\_RealType>

#### 5.652.1 Detailed Description

`template<typename _RealType = double> struct std::piecewise_linear_distribution<_RealType>::param_type`

Parameter type.

Definition at line 5109 of file random.h.

The documentation for this struct was generated from the following files:

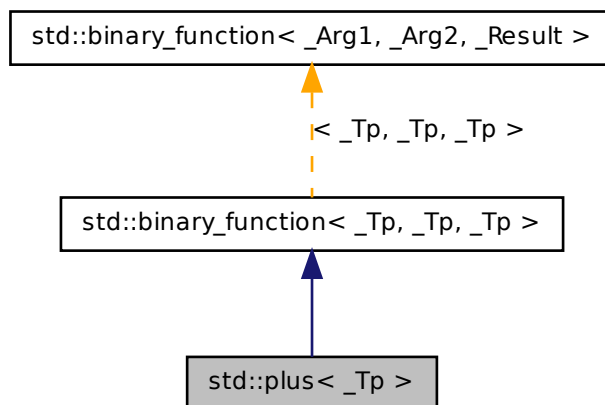
- [random.h](#)
- [random.tcc](#)



### 5.653 std::plus< \_Tp > Struct Template Reference

One of the [math functors](#).

Inheritance diagram for std::plus< \_Tp >:



#### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `_Tp` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

#### Public Member Functions

- `_Tp operator() (const _Tp &__x, const _Tp &__y) const`

#### 5.653.1 Detailed Description

**template<typename \_Tp> struct std::plus< \_Tp >**

One of the [math functors](#).

Definition at line 141 of file `stl_function.h`.

### 5.653.2 Member Typedef Documentation

**5.653.2.1** `typedef _Tp std::binary_function< _Tp , _Tp , _Tp  
>::first_argument_type [inherited]`

`first_argument_type` is the type of the first argument

Definition at line 118 of file `stl_function.h`.

**5.653.2.2** `typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::result_type  
[inherited]`

`result_type` is the return type

Definition at line 124 of file `stl_function.h`.

**5.653.2.3** `typedef _Tp std::binary_function< _Tp , _Tp , _Tp  
>::second_argument_type [inherited]`

`second_argument_type` is the type of the second argument

Definition at line 121 of file `stl_function.h`.

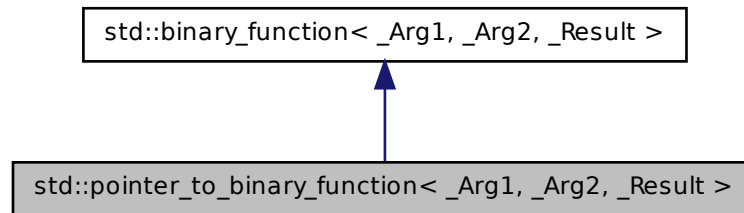
The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## **5.654 std::pointer\_to\_binary\_function< \_Arg1, \_Arg2, \_Result > Class Template Reference**

One of the [adaptors for function pointers](#).

Inheritance diagram for `std::pointer_to_binary_function< _Arg1, _Arg2, _Result >`:



### Public Types

- typedef `_Arg1` [first\\_argument\\_type](#)
- typedef `_Result` [result\\_type](#)
- typedef `_Arg2` [second\\_argument\\_type](#)

### Public Member Functions

- **`pointer_to_binary_function`** (`_Result(*__x)(_Arg1, _Arg2)`)
- `_Result operator()` (`_Arg1 __x, _Arg2 __y`) const

### Protected Attributes

- `_Result(* _M_ptr)(_Arg1, _Arg2)`

#### 5.654.1 Detailed Description

**template<typename `_Arg1`, typename `_Arg2`, typename `_Result`> class `std::pointer_to_binary_function< _Arg1, _Arg2, _Result >`**

One of the [adaptors for function pointers](#).

Definition at line 448 of file `stl_function.h`.

### 5.654.2 Member Typedef Documentation

**5.654.2.1** `template<typename _Arg1, typename _Arg2, typename _Result>  
typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result  
>::first_argument_type [inherited]`

`first_argument_type` is the type of the first argument

Definition at line 118 of file `stl_function.h`.

**5.654.2.2** `template<typename _Arg1, typename _Arg2, typename _Result>  
typedef _Result std::binary_function< _Arg1, _Arg2, _Result  
>::result_type [inherited]`

`result_type` is the return type

Definition at line 124 of file `stl_function.h`.

**5.654.2.3** `template<typename _Arg1, typename _Arg2, typename _Result>  
typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result  
>::second_argument_type [inherited]`

`second_argument_type` is the type of the second argument

Definition at line 121 of file `stl_function.h`.

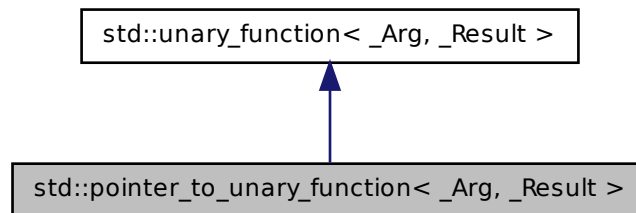
The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

### 5.655 std::pointer\_to\_unary\_function< \_Arg, \_Result > Class Template Reference

One of the [adaptors for function pointers](#).

Inheritance diagram for `std::pointer_to_unary_function< _Arg, _Result >`:



### Public Types

- typedef `_Arg` [argument\\_type](#)
- typedef `_Result` [result\\_type](#)

### Public Member Functions

- `pointer_to_unary_function` (`_Result`(\*\_\_x)(`_Arg`))
- `_Result operator()` (`_Arg __x`) const

### Protected Attributes

- `_Result(* _M_ptr)(_Arg)`

#### 5.655.1 Detailed Description

`template<typename _Arg, typename _Result> class std::pointer_to_unary_function< _Arg, _Result >`

One of the [adaptors for function pointers](#).

Definition at line 423 of file `stl_function.h`.

### 5.655.2 Member Typedef Documentation

**5.655.2.1** `template<typename _Arg, typename _Result> typedef _Arg  
std::unary_function< _Arg, _Result >::argument_type  
[inherited]`

`argument_type` is the type of the argument

Definition at line 105 of file `stl_function.h`.

**5.655.2.2** `template<typename _Arg, typename _Result> typedef _Result  
std::unary_function< _Arg, _Result >::result_type [inherited]`

`result_type` is the return type

Definition at line 108 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 5.656 `std::poisson_distribution< _IntType >` Class Template Reference

A discrete Poisson random number distribution.

### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_IntType` [result\\_type](#)

### Public Member Functions

- **`poisson_distribution`** (`double __mean=1.0`)
- **`poisson_distribution`** (`const param\_type &__p`)
- [result\\_type](#) **`max`** () const
- `double` [mean](#) () const

- `result_type min () const`
- `template<typename _UniformRandomNumberGenerator >  
result_type operator() (_UniformRandomNumberGenerator &__urng)`
- `template<typename _UniformRandomNumberGenerator >  
result_type operator() (_UniformRandomNumberGenerator &__urng, const  
param_type &__p)`
- `param_type param () const`
- `void param (const param_type &__param)`
- `void reset ()`

### Friends

- `template<typename _IntType1 , typename _CharT , typename _Traits >  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _  
CharT, _Traits > &, const std::poisson_distribution< _IntType1 > &)`
- `template<typename _IntType1 >  
bool operator== (const std::poisson_distribution< _IntType1 > &__d1, const  
std::poisson_distribution< _IntType1 > &__d2)`
- `template<typename _IntType1 , typename _CharT , typename _Traits >  
std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _  
CharT, _Traits > &, std::poisson_distribution< _IntType1 > &)`

#### 5.656.1 Detailed Description

`template<typename _IntType = int> class std::poisson_distribution< _IntType >`

A discrete Poisson random number distribution. The formula for the Poisson probability density function is  $p(i|\mu) = \frac{\mu^i}{i!} e^{-\mu}$  where  $\mu$  is the parameter of the distribution.

Definition at line 3950 of file random.h.

#### 5.656.2 Member Typedef Documentation

**5.656.2.1** `template<typename _IntType = int> typedef _IntType  
std::poisson_distribution< _IntType >::result_type`

The type of the range of the distribution.

Definition at line 3957 of file random.h.

### 5.656.3 Member Function Documentation

**5.656.3.1** `template<typename _IntType = int> result_type  
std::poisson_distribution<_IntType>::max ( ) const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 4044 of file `random.h`.

Referenced by `std::poisson_distribution<_IntType>::operator()()`.

**5.656.3.2** `template<typename _IntType = int> double  
std::poisson_distribution<_IntType>::mean ( ) const [inline]`

Returns the distribution parameter `mean`.

Definition at line 4015 of file `random.h`.

**5.656.3.3** `template<typename _IntType = int> result_type  
std::poisson_distribution<_IntType>::min ( ) const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 4037 of file `random.h`.

**5.656.3.4** `template<typename _IntType> template<typename  
_UniformRandomNumberGenerator> poisson_distribution<  
_IntType>::result_type std::poisson_distribution<_IntType  
>::operator() ( _UniformRandomNumberGenerator & __urng,  
const param_type & __param )`

A rejection algorithm when `mean >= 12` and a simple method based upon the multiplication of uniform random variates otherwise. NB: The former is available only if `_GLIBCXX_USE_C99_MATH_TR1` is defined.

Reference: Devroye, L. Non-Uniform Random Variates Generation. Springer-Verlag, New York, 1986, Ch. X, Sects. 3.3 & 3.4 (+ Errata!).

Definition at line 1200 of file `random.tcc`.

References `std::abs()`, `std::log()`, and `std::poisson_distribution<_IntType>::max()`.



**5.656.3.5** `template<typename _IntType = int> template<typename  
_UniformRandomNumberGenerator > result_type  
std::poisson_distribution< _IntType >::operator() (   
_UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 4052 of file random.h.

References `std::poisson_distribution< _IntType >::operator()()`, and `std::poisson_distribution< _IntType >::param()`.

Referenced by `std::poisson_distribution< _IntType >::operator()()`.

**5.656.3.6** `template<typename _IntType = int> param_type  
std::poisson_distribution< _IntType >::param ( ) const  
[inline]`

Returns the parameter set of the distribution.

Definition at line 4022 of file random.h.

Referenced by `std::poisson_distribution< _IntType >::operator()()`.

**5.656.3.7** `template<typename _IntType = int> void std::poisson_distribution<  
_IntType >::param ( const param_type & __param ) [inline]`

Sets the parameter set of the distribution.

#### Parameters

`__param` The new parameter set of the distribution.

Definition at line 4030 of file random.h.

**5.656.3.8** `template<typename _IntType = int> void std::poisson_distribution<  
_IntType >::reset ( ) [inline]`

Resets the distribution state.

Definition at line 4008 of file random.h.

References std::normal\_distribution< \_RealType >::reset().

#### 5.656.4 Friends And Related Function Documentation

**5.656.4.1** `template<typename _IntType = int> template<typename _IntType1 ,  
typename _CharT , typename _Traits > std::basic_ostream<_CharT,  
_Traits>& operator<< ( std::basic_ostream<_CharT, _Traits > & ,  
const std::poisson_distribution< _IntType1 > & ) [friend]`

Inserts a poisson\_distribution random number distribution `__x` into the output stream `__os`.

##### Parameters

`__os` An output stream.

`__x` A poisson\_distribution random number distribution.

##### Returns

The output stream with the state of `__x` inserted or in an error state.

**5.656.4.2** `template<typename _IntType = int> template<typename _IntType1  
> bool operator==( const std::poisson_distribution< _IntType1 >  
& __d1, const std::poisson_distribution< _IntType1 > & __d2 )  
[friend]`

Return true if two Poisson distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 4067 of file random.h.

**5.656.4.3** `template<typename _IntType = int> template<typename _IntType1 ,  
typename _CharT , typename _Traits > std::basic_istream<_CharT,  
_Traits>& operator>> ( std::basic_istream<_CharT, _Traits > & ,  
std::poisson_distribution< _IntType1 > & ) [friend]`

Extracts a poisson\_distribution random number distribution `__x` from the input stream `__is`.

## 5.657 `std::poisson_distribution< _IntType >::param_type` Struct Reference 3168

### Parameters

- `__is` An input stream.
- `__x` A `poisson_distribution` random number generator engine.

### Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.657 `std::poisson_distribution< _IntType >::param_type` Struct Reference

### Public Types

- typedef [poisson\\_distribution](#)< `_IntType` > `distribution_type`

### Public Member Functions

- `param_type` (double `__mean`=1.0)
- double `mean` () const

### Friends

- bool `operator==` (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- class `poisson_distribution`< `_IntType` >

### 5.657.1 Detailed Description

`template<typename _IntType = int> struct std::poisson_distribution< _IntType >::param_type`

Parameter type.

Definition at line 3959 of file `random.h`.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.658 `std::priority_queue< _Tp, _Sequence, _Compare >` Class Template Reference

A standard container automatically sorting its contents.

### Public Types

- typedef `_Sequence::const_reference` **const\_reference**
- typedef `_Sequence` **container\_type**
- typedef `_Sequence::reference` **reference**
- typedef `_Sequence::size_type` **size\_type**
- typedef `_Sequence::value_type` **value\_type**

### Public Member Functions

- [`priority\_queue`](#) (`const _Compare &__x, const _Sequence &__s`)
- **`priority_queue`** (`const _Compare &__x=_Compare(), _Sequence &&__s=_Sequence()`)
- template<typename `_InputIterator` >  
**`priority_queue`** (`_InputIterator __first, _InputIterator __last, const _Compare &__x=_Compare(), _Sequence &&__s=_Sequence()`)
- template<typename `_InputIterator` >  
[`priority\_queue`](#) (`_InputIterator __first, _InputIterator __last, const _Compare &__x, const _Sequence &__s`)
- template<typename... `_Args`>  
void **`emplace`** (`_Args &&...__args`)
- bool [`empty`](#) () const
- void [`pop`](#) ()
- void [`push`](#) (`const value_type &__x`)
- void **`push`** (`value_type &&__x`)
- size\_type [`size`](#) () const
- void **`swap`** ([`priority\_queue`](#) &\_\_pq)
- const\_reference [`top`](#) () const

### Protected Attributes

- `_Sequence` **`c`**
- `_Compare` **`comp`**

### 5.658.1 Detailed Description

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _
Compare = less<typename _Sequence::value_type>> class std::priority_queue<
_Tp, _Sequence, _Compare >
```

A standard container automatically sorting its contents. This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces priority-based sorting and queue behavior. Very few of the standard container/sequence interface requirements are met (e.g., iterators).

The second template parameter defines the type of the underlying sequence/container. It defaults to `std::vector`, but it can be any type that supports `front()`, `push_back`, `pop_back`, and random-access iterators, such as `std::deque` or an appropriate user-defined type.

The third template parameter supplies the means of making priority comparisons. It defaults to `less<value_type>` but can be anything defining a strict weak ordering.

Members not found in *normal* containers are `container_type`, which is a typedef for the second Sequence parameter, and `push`, `pop`, and `top`, which are standard queue operations.

#### Note

No equality/comparison operators are provided for `priority_queue`. Sorting of the elements takes place as they are added to, and removed from, the `priority_queue` using the `priority_queue`'s member functions. If you access the elements by other means, and change their data such that the sorting order would be different, the `priority_queue` will not re-sort the elements for you. (How could it know to do so?)

Definition at line 359 of file `stl_queue.h`.

### 5.658.2 Constructor & Destructor Documentation

```
5.658.2.1 template<typename _Tp, typename _Sequence = vector<_Tp>,
typename _Compare = less<typename _Sequence::value_type>>
std::priority_queue<_Tp, _Sequence, _Compare>::priority_queue (
const _Compare & __x, const _Sequence & __s) [inline,
explicit]
```

Default constructor creates no elements.

Definition at line 394 of file `stl_queue.h`.

References `std::make_heap()`.

```
5.658.2.2 template<typename _Tp, typename _Sequence = vector<_Tp>,
 typename _Compare = less<typename _Sequence::value_type>>
 template<typename _InputIterator > std::priority_queue<_Tp,
 _Sequence, _Compare>::priority_queue (_InputIterator __first,
 _InputIterator __last, const _Compare & __x, const _Sequence &
 __s) [inline]
```

Builds a queue from a range.

#### Parameters

- first* An input iterator.
- last* An input iterator.
- x* A comparison functor describing a strict weak ordering.
- s* An initial sequence with which to start.

Begins by copying *s*, inserting a copy of the elements from `[first,last)` into the copy of *s*, then ordering the copy according to *x*.

For more information on function objects, see the documentation on [functor base classes](#).

Definition at line 434 of file `stl_queue.h`.

References `std::make_heap()`.

#### 5.658.3 Member Function Documentation

```
5.658.3.1 template<typename _Tp, typename _Sequence = vector<_Tp>,
 typename _Compare = less<typename _Sequence::value_type>>
 bool std::priority_queue<_Tp, _Sequence, _Compare>::empty ()
 const [inline]
```

Returns true if the queue is empty.

Definition at line 460 of file `stl_queue.h`.

Referenced by `__gnu_parallel::multiseq_partition()`, and `__gnu_parallel::multiseq_selection()`.

**5.658.3.2** `template<typename _Tp, typename _Sequence = vector<_Tp>,  
typename _Compare = less<typename _Sequence::value_type>>  
void std::priority_queue<_Tp, _Sequence, _Compare >::pop ( )  
[inline]`

Removes first element.

This is a typical queue operation. It shrinks the queue by one. The time complexity of the operation depends on the underlying sequence.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop()` is called.

Definition at line 523 of file `stl_queue.h`.

References `std::pop_heap()`.

Referenced by `__gnu_parallel::multiseq_partition()`, and `__gnu_parallel::multiseq_selection()`.

**5.658.3.3** `template<typename _Tp, typename _Sequence = vector<_Tp>,  
typename _Compare = less<typename _Sequence::value_type>>  
void std::priority_queue<_Tp, _Sequence, _Compare >::push (  
const value_type & __x ) [inline]`

Add data to the queue.

#### Parameters

*x* Data to be added.

This is a typical queue operation. The time complexity of the operation depends on the underlying sequence.

Definition at line 488 of file `stl_queue.h`.

References `std::push_heap()`.

Referenced by `__gnu_parallel::multiseq_partition()`, and `__gnu_parallel::multiseq_selection()`.

**5.658.3.4** `template<typename _Tp, typename _Sequence = vector<_Tp>,  
typename _Compare = less<typename _Sequence::value_type>>  
size_type std::priority_queue<_Tp, _Sequence, _Compare>::size(  
) const [inline]`

Returns the number of elements in the queue.

Definition at line 465 of file `stl_queue.h`.

**5.658.3.5** `template<typename _Tp, typename _Sequence = vector<_Tp>,  
typename _Compare = less<typename _Sequence::value_type>>  
const_reference std::priority_queue<_Tp, _Sequence, _Compare  
>::top() const [inline]`

Returns a read-only (constant) reference to the data at the first element of the queue.

Definition at line 473 of file `stl_queue.h`.

Referenced by `__gnu_parallel::multiseq_partition()`, and `__gnu_parallel::multiseq_selection()`.

The documentation for this class was generated from the following file:

- [stl\\_queue.h](#)

## 5.659 `std::promise<_Res>` Class Template Reference

Primary template for promise.

### Public Member Functions

- `promise` ([promise](#) &&\_\_rhs)
- `promise` (const [promise](#) &)
- `template<typename _Allocator>  
promise` ([allocator\\_arg\\_t](#), const `_Allocator` &\_\_a)
- `future<_Res>` `get_future` ()
- `promise` & `operator=` (const [promise](#) &)
- `promise` & `operator=` ([promise](#) &&\_\_rhs)
- void `set_exception` (exception\_ptr \_\_p)
- void `set_value` (`_Res` &&\_\_r)
- void `set_value` (const `_Res` &\_\_r)
- void `swap` ([promise](#) &&\_\_rhs)



**Friends**

- `class _State::_Setter`

**5.659.1 Detailed Description**

`template<typename _Res> class std::promise<_Res >`

Primary template for promise.

Definition at line 851 of file future.

The documentation for this class was generated from the following file:

- [future](#)

**5.660 `std::promise<_Res &>` Class Template Reference**

Partial specialization for `promise<R&>`

**Public Member Functions**

- `promise` ([promise](#) &&\_\_rhs)
- `promise` (const [promise](#) &)
- `template<typename _Allocator > promise` ([allocator\\_arg\\_t](#), const `_Allocator &`\_\_a)
- `future<_Res &> get_future` ()
- `promise & operator=` (const [promise](#) &)
- `promise & operator=` ([promise](#) &&\_\_rhs)
- `void set_exception` (exception\_ptr \_\_p)
- `void set_value` (`_Res &`\_\_r)
- `void swap` ([promise](#) &\_\_rhs)

**Friends**

- `class _State::_Setter`

**5.660.1 Detailed Description**

`template<typename _Res> class std::promise<_Res &>`

Partial specialization for `promise<R&>`

Definition at line 943 of file future.

The documentation for this class was generated from the following file:

- [future](#)

## 5.661 `std::promise< void >` Class Template Reference

Explicit specialization for `promise<void>`

### Public Member Functions

- `promise` (`promise` &&\_\_rhs)
- `promise` (const `promise` &)
- `template<typename _Allocator > promise` (`allocator_arg_t`, const `_Allocator` &\_\_a)
- `future< void > get_future` ()
- `promise` & `operator=` (const `promise` &)
- `promise` & `operator=` (`promise` &&\_\_rhs)
- void `set_exception` (`exception_ptr` \_\_p)
- void `set_value` ()
- void `swap` (`promise` &\_\_rhs)

### Friends

- class `_State::_Setter`

#### 5.661.1 Detailed Description

`template<> class std::promise< void >`

Explicit specialization for `promise<void>`

Definition at line 1018 of file future.

The documentation for this class was generated from the following file:

- [future](#)

## 5.662 `std::queue< _Tp, _Sequence >` Class Template Reference

A standard container giving FIFO behavior.

### Public Types

- `typedef _Sequence::const_reference` **const\_reference**
- `typedef _Sequence` **container\_type**
- `typedef _Sequence::reference` **reference**
- `typedef _Sequence::size_type` **size\_type**
- `typedef _Sequence::value_type` **value\_type**

### Public Member Functions

- `queue` (const \_Sequence &\_\_c)
- **queue** (\_Sequence &&\_\_c=\_Sequence())
- const\_reference `back` () const
- reference `back` ()
- `template<typename... _Args>`  
void **emplace** (\_Args &&...\_\_args)
- bool `empty` () const
- reference `front` ()
- const\_reference `front` () const
- void `pop` ()
- void **push** (value\_type &&\_\_x)
- void `push` (const value\_type &\_\_x)
- size\_type `size` () const
- void **swap** (queue &\_\_q)

### Protected Attributes

- \_Sequence `c`

### Friends

- `template<typename _Tp1, typename _Seq1 >`  
bool **operator**< (const queue<\_Tp1, \_Seq1 > &, const queue<\_Tp1, \_Seq1 > &)
- `template<typename _Tp1, typename _Seq1 >`  
bool **operator**== (const queue<\_Tp1, \_Seq1 > &, const queue<\_Tp1, \_Seq1 > &)

### 5.662.1 Detailed Description

**template<typename \_Tp, typename \_Sequence = deque<\_Tp>> class std::queue<\_Tp, \_Sequence>**

A standard container giving FIFO behavior. Meets many of the requirements of a `container`, but does not define anything to do with iterators. Very few of the other standard container interfaces are defined.

This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces strict first-in-first-out queue behavior.

The second template parameter defines the type of the underlying sequence/container. It defaults to `std::deque`, but it can be any type that supports `front`, `back`, `push_back`, and `pop_front`, such as `std::list` or an appropriate user-defined type.

Members not found in *normal* containers are `container_type`, which is a typedef for the second `Sequence` parameter, and `push` and `pop`, which are standard queue/FIFO operations.

Definition at line 92 of file `stl_queue.h`.

### 5.662.2 Constructor & Destructor Documentation

**5.662.2.1** **template<typename \_Tp, typename \_Sequence = deque<\_Tp>> std::queue<\_Tp, \_Sequence>::queue ( const \_Sequence & \_\_c ) [inline, explicit]**

Default constructor creates no elements.

Definition at line 137 of file `stl_queue.h`.

### 5.662.3 Member Function Documentation

**5.662.3.1** **template<typename \_Tp, typename \_Sequence = deque<\_Tp>> reference std::queue<\_Tp, \_Sequence>::back ( ) [inline]**

Returns a read/write reference to the data at the last element of the queue.

Definition at line 184 of file `stl_queue.h`.

References `std::queue<_Tp, _Sequence>::c`.

**5.662.3.2** `template<typename _Tp, typename _Sequence = deque<_Tp>>  
const_reference std::queue<_Tp, _Sequence>::back ( ) const  
[inline]`

Returns a read-only (constant) reference to the data at the last element of the queue.

Definition at line 195 of file `stl_queue.h`.

References `std::queue<_Tp, _Sequence>::c`.

**5.662.3.3** `template<typename _Tp, typename _Sequence = deque<_Tp>>  
bool std::queue<_Tp, _Sequence>::empty ( ) const [inline]`

Returns true if the queue is empty.

Definition at line 149 of file `stl_queue.h`.

References `std::queue<_Tp, _Sequence>::c`.

**5.662.3.4** `template<typename _Tp, typename _Sequence = deque<_Tp>>  
const_reference std::queue<_Tp, _Sequence>::front ( ) const  
[inline]`

Returns a read-only (constant) reference to the data at the first element of the queue.

Definition at line 173 of file `stl_queue.h`.

References `std::queue<_Tp, _Sequence>::c`.

**5.662.3.5** `template<typename _Tp, typename _Sequence = deque<_Tp>>  
reference std::queue<_Tp, _Sequence>::front ( ) [inline]`

Returns a read/write reference to the data at the first element of the queue.

Definition at line 162 of file `stl_queue.h`.

References `std::queue<_Tp, _Sequence>::c`.

**5.662.3.6** `template<typename _Tp, typename _Sequence = deque<_Tp>>  
void std::queue<_Tp, _Sequence>::pop ( ) [inline]`

Removes first element.

This is a typical queue operation. It shrinks the queue by one. The time complexity of the operation depends on the underlying sequence.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop()` is called.

Definition at line 237 of file `stl_queue.h`.

References `std::queue<_Tp, _Sequence>::c`.

**5.662.3.7** `template<typename _Tp, typename _Sequence = deque<_Tp>>`  
`void std::queue<_Tp, _Sequence>::push ( const value_type & __x`  
`) [inline]`

Add data to the end of the queue.

#### Parameters

*x* Data to be added.

This is a typical queue operation. The function creates an element at the end of the queue and assigns the given data to it. The time complexity of the operation depends on the underlying sequence.

Definition at line 211 of file `stl_queue.h`.

References `std::queue<_Tp, _Sequence>::c`.

**5.662.3.8** `template<typename _Tp, typename _Sequence = deque<_Tp>>`  
`size_type std::queue<_Tp, _Sequence>::size ( ) const [inline]`

Returns the number of elements in the queue.

Definition at line 154 of file `stl_queue.h`.

References `std::queue<_Tp, _Sequence>::c`.

#### 5.662.4 Member Data Documentation

**5.662.4.1** `template<typename _Tp, typename _Sequence = deque<_Tp>>`  
`_Sequence std::queue<_Tp, _Sequence>::c [protected]`

'c' is the underlying container. Maintainers wondering why this isn't uglified as per style guidelines should note that this name is specified in the standard, [23.2.3.1]. (Why? Presumably for the same reason that it's protected instead of private: to allow derivation. But none of the other containers allow for derivation. Odd.)

Definition at line 125 of file `stl_queue.h`.

Referenced by `std::queue< _Tp, _Sequence >::back()`, `std::queue< _Tp, _Sequence >::empty()`, `std::queue< _Tp, _Sequence >::front()`, `std::operator==( )`, `std::queue< _Tp, _Sequence >::pop()`, `std::queue< _Tp, _Sequence >::push()`, and `std::queue< _Tp, _Sequence >::size()`.

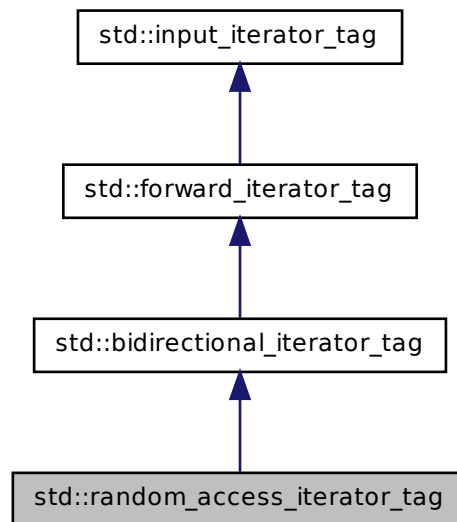
The documentation for this class was generated from the following file:

- [stl\\_queue.h](#)

### 5.663 `std::random_access_iterator_tag` Struct Reference

Random-access iterators support a superset of bidirectional /// iterator operations.

Inheritance diagram for `std::random_access_iterator_tag`:



#### 5.663.1 Detailed Description

Random-access iterators support a superset of bidirectional /// iterator operations.

Definition at line 104 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 5.664 `std::random_device` Class Reference

### Public Types

- typedef unsigned int [result\\_type](#)

### Public Member Functions

- **random\_device** (const [std::string](#) &\_\_token="/dev/urandom")
- **random\_device** (const [random\\_device](#) &)
- double **entropy** () const
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- [result\\_type](#) **operator**() ()
- void **operator=** (const [random\\_device](#) &)

#### 5.664.1 Detailed Description

A standard interface to a platform-specific non-deterministic random number generator (if any are available).

Definition at line 1498 of file `random.h`.

#### 5.664.2 Member Typedef Documentation

##### 5.664.2.1 typedef unsigned int `std::random_device::result_type`

The type of the generated random value.

Definition at line 1502 of file `random.h`.

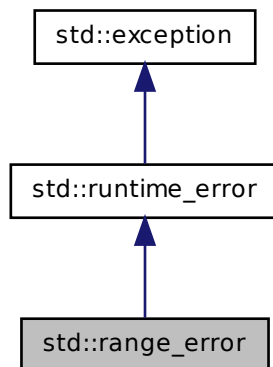
The documentation for this class was generated from the following file:

- [random.h](#)



## 5.665 `std::range_error` Class Reference

Inheritance diagram for `std::range_error`:



### Public Member Functions

- **`range_error`** (const [string](#) &\_\_arg)
- virtual const char \* **`what`** () const throw ()

#### 5.665.1 Detailed Description

Thrown to indicate range errors in internal computations.

Definition at line 129 of file `stdexcept`.

#### 5.665.2 Member Function Documentation

##### 5.665.2.1 virtual const char\* `std::runtime_error::what` ( ) const throw () [**virtual**, **inherited**]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

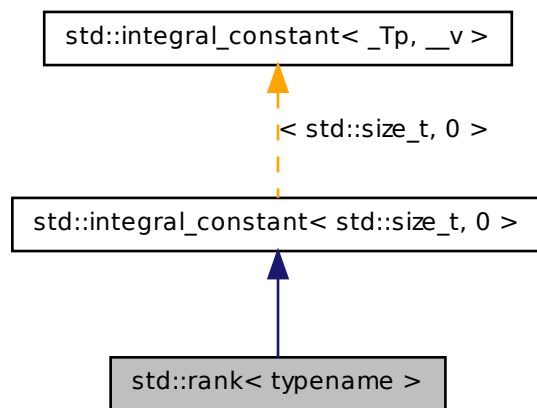
The documentation for this class was generated from the following file:

- [stdexcept](#)

## 5.666 std::rank< typename > Struct Template Reference

rank

Inheritance diagram for std::rank< typename >:



### Public Types

- typedef [integral\\_constant](#)< std::size\_t, \_\_v > **type**
- typedef std::size\_t **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr std::size\_t **value**

### 5.666.1 Detailed Description

**template<typename> struct std::rank< typename >**

rank

Definition at line 372 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.667 `std::ratio<_Num, _Den >` Struct Template Reference

Provides compile-time rational arithmetic.

### Public Types

- typedef [ratio](#)< num, den > **type**

### Public Member Functions

- **static\_assert** (\_Den!=0,"denominator cannot be zero")
- **static\_assert** (\_Num >= \_\_INTMAX\_MAX\_\_ && \_Den >= \_\_INTMAX\_MAX\_\_, "out of range")

### Static Public Attributes

- static constexpr intmax\_t **den**
- static constexpr intmax\_t **num**

### 5.667.1 Detailed Description

**template<intmax\_t \_Num, intmax\_t \_Den = 1> struct std::ratio< \_Num, \_Den >**

Provides compile-time rational arithmetic. This class template represents any finite rational number with a numerator and denominator representable by compile-time constants of type `intmax_t`. The ratio is simplified when instantiated.

For example:

```
std::ratio<7,-21>::num == -1;
std::ratio<7,-21>::den == 3;
```

Definition at line 153 of file `ratio`.

The documentation for this struct was generated from the following file:

- [ratio](#)

## 5.668 `std::ratio_add< _R1, _R2 >` Struct Template Reference

[ratio\\_add](#)

### Public Types

- typedef [ratio](#)< `__safe_add< __safe_multiply< _R1::num,(_R2::den/___gcd)>::value, __safe_multiply< _R2::num,(_R1::den/___gcd)>::value >::value, __safe_multiply< _R1::den,(_R2::den/___gcd)>::value >` **type**

### Static Public Attributes

- static constexpr intmax\_t **den**
- static constexpr intmax\_t **num**

#### 5.668.1 Detailed Description

`template<typename _R1, typename _R2> struct std::ratio_add< _R1, _R2 >`

[ratio\\_add](#)

Definition at line 177 of file `ratio`.

The documentation for this struct was generated from the following file:

- [ratio](#)

## 5.669 `std::ratio_divide< _R1, _R2 >` Struct Template Reference

[ratio\\_divide](#)

### Public Types

- typedef [ratio\\_multiply](#)< `_R1, ratio< _R2::den, _R2::num > >::type` **type**

**Public Member Functions**

- `static_assert` (`_R2::num!=0`, "division by 0")

**Static Public Attributes**

- static constexpr intmax\_t **den**
- static constexpr intmax\_t **num**

**5.669.1 Detailed Description**

`template<typename _R1, typename _R2> struct std::ratio_divide< _R1, _R2 >`

[ratio\\_divide](#)

Definition at line 247 of file `ratio`.

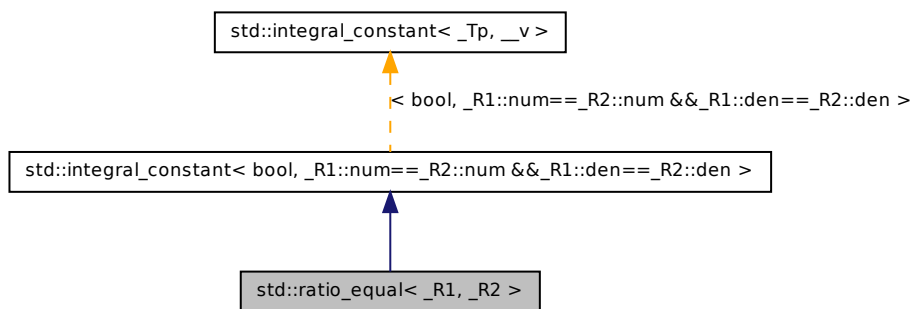
The documentation for this struct was generated from the following file:

- [ratio](#)

**5.670 `std::ratio_equal< _R1, _R2 >` Struct Template Reference**

[ratio\\_equal](#)

Inheritance diagram for `std::ratio_equal< _R1, _R2 >`:



### Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr bool **value**

#### 5.670.1 Detailed Description

`template<typename _R1, typename _R2> struct std::ratio_equal<_R1,_R2>`

[ratio\\_equal](#)

Definition at line 267 of file `ratio`.

The documentation for this struct was generated from the following file:

- [ratio](#)

### 5.671 `std::ratio_multiply<_R1,_R2>` Struct Template Reference

[ratio\\_multiply](#)

### Public Types

- typedef [ratio](#)< `__safe_multiply<(_R1::num/__gcd1),(_R2::num/_  
__gcd2)>::value,` `__safe_multiply<(_R1::den/__gcd2),(_R2::den/_  
gcd1)>::value` > **type**

### Static Public Attributes

- static constexpr intmax\_t **den**
- static constexpr intmax\_t **num**

## 5.671.1 Detailed Description

```
template<typename _R1, typename _R2> struct std::ratio_multiply< _R1, _R2
>
```

[ratio\\_multiply](#)

Definition at line 220 of file `ratio`.

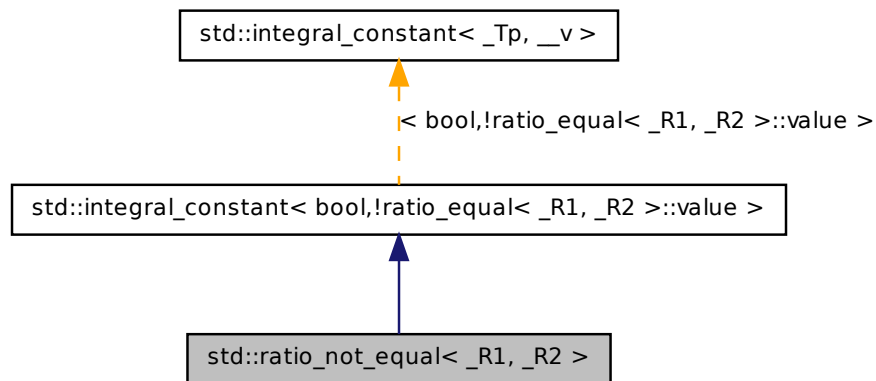
The documentation for this struct was generated from the following file:

- [ratio](#)

5.672 `std::ratio_not_equal< _R1, _R2 >` Struct Template Reference

[ratio\\_not\\_equal](#)

Inheritance diagram for `std::ratio_not_equal< _R1, _R2 >`:



## Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

### Public Member Functions

- `constexpr operator value_type ()`

### Static Public Attributes

- `static constexpr bool value`

#### 5.672.1 Detailed Description

`template<typename _R1, typename _R2> struct std::ratio_not_equal< _R1, _R2 >`

[ratio\\_not\\_equal](#)

Definition at line 273 of file `ratio`.

The documentation for this struct was generated from the following file:

- [ratio](#)

### 5.673 `std::ratio_subtract< _R1, _R2 >` Struct Template Reference

[ratio\\_subtract](#)

### Public Types

- `typedef ratio\_add< \_R1, ratio<-\\_R2::num, \\_R2::den > >::type type`

### Static Public Attributes

- `static constexpr intmax_t den`
- `static constexpr intmax_t num`

#### 5.673.1 Detailed Description

`template<typename _R1, typename _R2> struct std::ratio_subtract< _R1, _R2 >`

[ratio\\_subtract](#)

Definition at line 202 of file `ratio`.

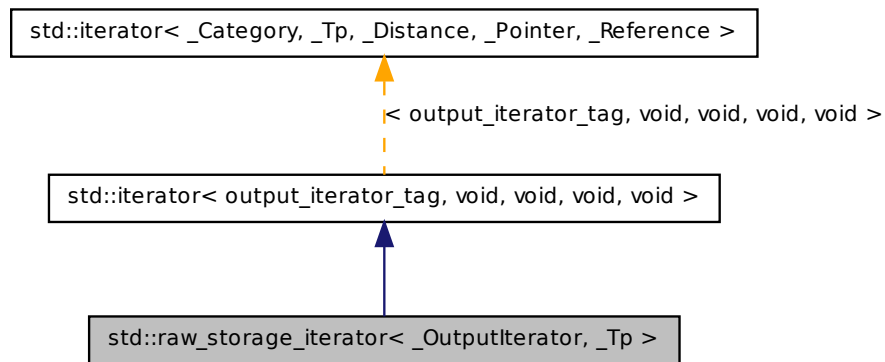
The documentation for this struct was generated from the following file:



- [ratio](#)

## 5.674 `std::raw_storage_iterator< _OutputIterator, _Tp >` Class Template Reference

Inheritance diagram for `std::raw_storage_iterator< _OutputIterator, _Tp >`:



### Public Types

- typedef void [difference\\_type](#)
- typedef [output\\_iterator\\_tag](#) [iterator\\_category](#)
- typedef void [pointer](#)
- typedef void [reference](#)
- typedef void [value\\_type](#)

### Public Member Functions

- **`raw_storage_iterator`** (`_OutputIterator __x`)
- [raw\\_storage\\_iterator](#) & **`operator*`** ()
- [raw\\_storage\\_iterator](#)< `_OutputIterator, _Tp` > & **`operator++`** ()
- [raw\\_storage\\_iterator](#)< `_OutputIterator, _Tp` > **`operator++`** (int)
- [raw\\_storage\\_iterator](#) & **`operator=`** (const `_Tp` & `__element`)

### Protected Attributes

- `_OutputIterator _M_iter`

#### 5.674.1 Detailed Description

**template<class \_OutputIterator, class \_Tp> class std::raw\_storage\_iterator< \_OutputIterator, \_Tp >**

This iterator class lets algorithms store their results into uninitialized memory.

Definition at line 69 of file `stl_raw_storage_iter.h`.

#### 5.674.2 Member Typedef Documentation

**5.674.2.1 typedef void std::iterator< output\_iterator\_tag , void , void , void , void >::difference\_type [inherited]**

Distance between iterators is represented as this type.

Definition at line 126 of file `stl_iterator_base_types.h`.

**5.674.2.2 typedef output\_iterator\_tag std::iterator< output\_iterator\_tag , void , void , void , void >::iterator\_category [inherited]**

One of the [tag types](#).

Definition at line 122 of file `stl_iterator_base_types.h`.

**5.674.2.3 typedef void std::iterator< output\_iterator\_tag , void , void , void , void >::pointer [inherited]**

This type represents a pointer-to-value\_type.

Definition at line 128 of file `stl_iterator_base_types.h`.

**5.674.2.4 typedef void std::iterator< output\_iterator\_tag , void , void , void , void >::reference [inherited]**

This type represents a reference-to-value\_type.

Definition at line 130 of file `stl_iterator_base_types.h`.

**5.674.2.5** `typedef void std::iterator< output_iterator_tag , void , void , void , void >::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 124 of file `stl_iterator_base_types.h`.

The documentation for this class was generated from the following file:

- [stl\\_raw\\_storage\\_iter.h](#)

## 5.675 `std::recursive_mutex` Class Reference

[recursive\\_mutex](#)

### Public Types

- `typedef __native_type * native_handle_type`

### Public Member Functions

- `recursive_mutex` (const [recursive\\_mutex](#) &)
- `void lock` ()
- `native_handle_type native_handle` ()
- `recursive\_mutex & operator=` (const [recursive\\_mutex](#) &)
- `bool try_lock` ()
- `void unlock` ()

### 5.675.1 Detailed Description

[recursive\\_mutex](#)

Definition at line 156 of file `mutex`.

The documentation for this class was generated from the following file:

- [mutex](#)

## 5.676 `std::recursive_timed_mutex` Class Reference

[recursive\\_timed\\_mutex](#)

### Public Types

- typedef `__native_type * native_handle_type`

### Public Member Functions

- `recursive_timed_mutex` (const [recursive\\_timed\\_mutex](#) &)
- void `lock` ()
- `native_handle_type native_handle` ()
- [recursive\\_timed\\_mutex](#) & `operator=` (const [recursive\\_timed\\_mutex](#) &)
- bool `try_lock` ()
- template<class `_Rep` , class `_Period` >  
bool `try_lock_for` (const [chrono::duration](#)< `_Rep`, `_Period` > &\_\_rtime)
- template<class `_Clock` , class `_Duration` >  
bool `try_lock_until` (const [chrono::time\\_point](#)< `_Clock`, `_Duration` > &\_\_atime)
- void `unlock` ()

### 5.676.1 Detailed Description

[recursive\\_timed\\_mutex](#)

Definition at line 315 of file `mutex`.

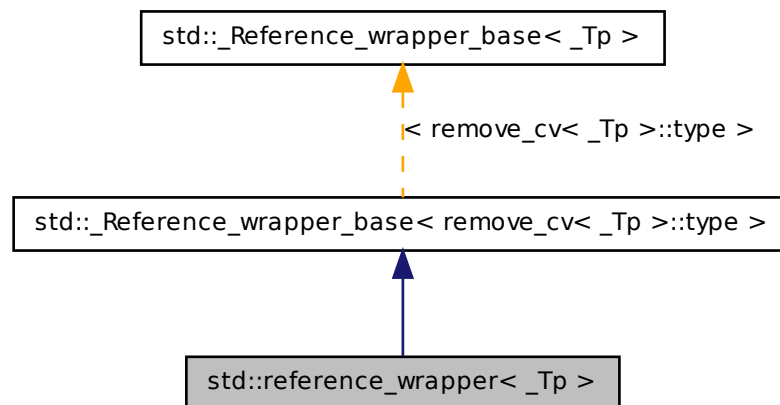
The documentation for this class was generated from the following file:

- [mutex](#)

## 5.677 `std::reference_wrapper<_Tp>` Class Template Reference

Primary class template for [reference\\_wrapper](#).

Inheritance diagram for std::reference\_wrapper< \_Tp >:



### Public Types

- `typedef _Tp type`

### Public Member Functions

- **reference\_wrapper** (\_Tp &\_\_indata)
- **reference\_wrapper** (\_Tp &&)
- **reference\_wrapper** (const [reference\\_wrapper](#)< \_Tp > &\_\_inref)
- \_Tp & **get** () const
- **operator \_Tp &** () const
- `template<typename... _Args>`  
`result_of< _M_func_type(_Args...)>::type operator() (_Args &&...__args)`  
`const`
- [reference\\_wrapper](#) & **operator=** (const [reference\\_wrapper](#)< \_Tp > &\_\_inref)

#### 5.677.1 Detailed Description

`template<typename _Tp> class std::reference_wrapper< _Tp >`

Primary class template for [reference\\_wrapper](#).

Definition at line 422 of file `functional`.

The documentation for this class was generated from the following file:

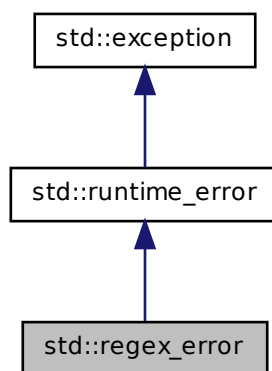
- [functional](#)

## 5.678 `std::regex_error` Class Reference

A regular expression exception class.

The regular expression library throws objects of this class on error.

Inheritance diagram for `std::regex_error`:



### Public Member Functions

- [regex\\_error](#) ([regex\\_constants::error\\_type](#) \_\_ecode)
- [regex\\_constants::error\\_type](#) code () const
- virtual const char \* [what](#) () const throw ()

### Protected Attributes

- [regex\\_constants::error\\_type](#) \_M\_code

### 5.678.1 Detailed Description

A regular expression exception class.

The regular expression library throws objects of this class on error.

Definition at line 131 of file `regex_error.h`.

### 5.678.2 Constructor & Destructor Documentation

**5.678.2.1** `std::regex_error::regex_error ( regex_constants::error_type __ecode ) [inline, explicit]`

Constructs a `regex_error` object.

#### Parameters

*ecode* the regex error code.

Definition at line 141 of file `regex_error.h`.

### 5.678.3 Member Function Documentation

**5.678.3.1** `regex_constants::error_type std::regex_error::code ( ) const [inline]`

Gets the regex error code.

#### Returns

the regex error code.

Definition at line 151 of file `regex_error.h`.

**5.678.3.2** `virtual const char* std::runtime_error::what ( ) const throw () [virtual, inherited]`

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from `std::exception`.

The documentation for this class was generated from the following file:

- [regex\\_error.h](#)

## 5.679 `std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>` Class Template Reference

### Public Types

- typedef `std::ptrdiff_t` **difference\_type**
- typedef `std::forward_iterator_tag` **iterator\_category**
- typedef const **value\_type** \* **pointer**
- typedef const **value\_type** & **reference**
- typedef `basic_regex<_Ch_type, _Rx_traits>` **regex\_type**
- typedef `match_results<_Bi_iter>` **value\_type**

### Public Member Functions

- [regex\\_iterator](#) ()
- [regex\\_iterator](#) (`_Bi_iter __a, _Bi_iter __b, const regex_type &__re, regex_constants::match_flag_type __m=regex_constants::match_default`)
- [regex\\_iterator](#) (`const regex_iterator &__rhs`)
- `bool operator!=` (`const regex_iterator &__rhs`)
- `const value_type & operator*` ()
- `regex_iterator & operator++` ()
- `regex_iterator operator++` (`int`)
- `const value_type * operator->` ()
- `regex_iterator & operator=` (`const regex_iterator &__rhs`)
- `bool operator==` (`const regex_iterator &__rhs`)

#### 5.679.1 Detailed Description

`template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> class std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>`

An iterator adaptor that will provide repeated calls of `regex_search` over a range until no more matches remain.

Definition at line 2149 of file `regex.h`.



## 5.679.2 Constructor & Destructor Documentation

**5.679.2.1** `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::regex_iterator ( )`

Provides a singular iterator, useful for indicating one-past-the-end of a range.

### Todo

Implement this function.

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

**5.679.2.2** `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::regex_iterator ( _Bi_iter __a, _Bi_iter __b, const regex_type & __re, regex_constants::match_flag_type __m = regex_constants::match_default )`

Constructs a `regex_iterator`...

### Parameters

*a* [IN] The start of a text range to search.

*b* [IN] One-past-the-end of the text range to search.

*re* [IN] The regular expression to match.

*m* [IN] Policy flags for match rules.

### Todo

Implement this function.

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

**5.679.2.3** `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::regex_iterator ( const regex_iterator<_Bi_iter, _Ch_type, _Rx_traits> & __rhs )`

Copy constructs a `regex_iterator`.

**Todo**

Implement this function.

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

**5.679.3 Member Function Documentation**

**5.679.3.1** `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> bool std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator!=( const regex_iterator<_Bi_iter, _Ch_type, _Rx_traits> & __rhs )`

**Todo**

Implement this function.

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

**5.679.3.2** `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> const value_type& std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator* ( )`

**Todo**

Implement this function.

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

**5.679.3.3** `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> regex_iterator std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator++ ( int )`

**Todo**

Implement this function.

**5.679 std::regex\_iterator< \_Bi\_iter, \_Ch\_type, \_Rx\_traits > Class Template Reference 3200**

---

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

**5.679.3.4** `template<typename _Bi_iter, typename _Ch_type = typename  
iterator_traits<_Bi_iter>::value_type, typename _Rx_traits =  
regex_traits<_Ch_type>> regex_iterator& std::regex_iterator<  
_Bi_iter, _Ch_type, _Rx_traits >::operator++ ( )`

**Todo**

Implement this function.

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

**5.679.3.5** `template<typename _Bi_iter, typename _Ch_type = typename  
iterator_traits<_Bi_iter>::value_type, typename _Rx_traits =  
regex_traits<_Ch_type>> const value_type* std::regex_iterator<  
_Bi_iter, _Ch_type, _Rx_traits >::operator-> ( )`

**Todo**

Implement this function.

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

**5.679.3.6** `template<typename _Bi_iter, typename _Ch_type = typename  
iterator_traits<_Bi_iter>::value_type, typename _Rx_traits =  
regex_traits<_Ch_type>> regex_iterator& std::regex_iterator<  
_Bi_iter, _Ch_type, _Rx_traits >::operator= ( const regex_iterator<  
_Bi_iter, _Ch_type, _Rx_traits > & __rhs )`

**Todo**

Implement this function.

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

5.679.3.7 `template<typename _Bi_iter, typename _Ch_type = typename  
iterator_traits<_Bi_iter>::value_type, typename _Rx_traits =  
regex_traits<_Ch_type>> bool std::regex_iterator< _Bi_iter,  
_Ch_type, _Rx_traits >::operator==( const regex_iterator<  
_Bi_iter, _Ch_type, _Rx_traits > & __rhs )`

### Todo

Implement this function.

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

The documentation for this class was generated from the following file:

- [regex.h](#)

## 5.680 `std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >` Class Template Reference

### Public Types

- `typedef std::ptrdiff_t difference_type`
- `typedef std::forward\_iterator\_tag iterator_category`
- `typedef const value\_type * pointer`
- `typedef const value\_type & reference`
- `typedef basic\_regex< _Ch_type, _Rx_traits > regex_type`
- `typedef sub\_match< _Bi_iter > value_type`

### Public Member Functions

- `regex\_token\_iterator ()`
- `regex\_token\_iterator (_Bi_iter __a, _Bi_iter __b, const regex\_type &__re, int __submatch=0, regex\_constants::match\_flag\_type __m=regex\_constants::match\_default)`
- `template<std::size_t _Nm>  
regex\_token\_iterator (_Bi_iter __a, _Bi_iter __b, const regex\_type &__re, const int(&__submatches)[_Nm], regex\_constants::match\_flag\_type __m=regex\_constants::match\_default)`
- `regex\_token\_iterator (const regex\_token\_iterator &__rhs)`
- `regex\_token\_iterator (_Bi_iter __a, _Bi_iter __b, const regex\_type &__re, const std::vector< int > &__submatches, regex\_constants::match\_flag\_type __m=regex\_constants::match\_default)`

- `bool operator!= (const regex\_token\_iterator &__rhs)`
- `const value\_type & operator* ()`
- `regex\_token\_iterator & operator++ ()`
- `regex\_token\_iterator operator++ (int)`
- `const value\_type * operator-> ()`
- `regex\_token\_iterator & operator= (const regex\_token\_iterator &__rhs)`
- `bool operator== (const regex\_token\_iterator &__rhs)`

### 5.680.1 Detailed Description

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_
_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>>> class
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >
```

Iterates over submatches in a range (or *splits* a text string).

The purpose of this iterator is to enumerate all, or all specified, matches of a regular expression within a text range. The dereferenced value of an iterator of this class is a [std::sub\\_match](#) object.

Definition at line 2264 of file `regex.h`.

### 5.680.2 Constructor & Destructor Documentation

**5.680.2.1** `template<typename _Bi_iter , typename _Ch_type = typename  
iterator_traits<_Bi_iter>::value_type, typename _Rx_traits =  
regex\_traits<\_Ch\_type>>> std::regex_token_iterator< _Bi_iter,  
_Ch_type, _Rx_traits >::regex_token_iterator ( )`

Default constructs a `regex_token_iterator`.

#### [Todo](#)

Implement this function.

A default-constructed `regex_token_iterator` is a singular iterator that will compare equal to the one-past-the-end value for any iterator of the same type.

5.680.2.2 `template<typename _Bi_iter , typename _Ch_type = typename  
iterator_traits<_Bi_iter>::value_type, typename _Rx_traits  
= regex_traits<_Ch_type>> std::regex_token_iterator<  
_Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator (   
_Bi_iter __a, _Bi_iter __b, const regex_type & __re, int  
__submatch = 0, regex_constants::match_flag_type __m =  
regex_constants::match_default )`

Constructs a regex\_token\_iterator...

#### Parameters

*a* [IN] The start of the text to search.

*b* [IN] One-past-the-end of the text to search.

*re* [IN] The regular expression to search for.

*submatch* [IN] Which submatch to return. There are some special values for this parameter:

- -1 each enumerated subexpression does NOT match the regular expression (aka field splitting)
- 0 the entire string matching the subexpression is returned for each match within the text.
- >0 enumerates only the indicated subexpression from a match within the text.

*m* [IN] Policy flags for match rules.

#### Todo

Implement this function.

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

5.680.2.3 `template<typename _Bi_iter , typename _Ch_type = typename  
iterator_traits<_Bi_iter>::value_type, typename _Rx_traits =  
regex_traits<_Ch_type>> std::regex_token_iterator< _Bi_iter,  
_Ch_type, _Rx_traits >::regex_token_iterator ( _Bi_iter __a,  
_Bi_iter __b, const regex_type & __re, const std::vector< int  
> & __submatches, regex_constants::match_flag_type __m =  
regex_constants::match_default )`

Constructs a regex\_token\_iterator...

#### Parameters

*a* [IN] The start of the text to search.

*b* [IN] One-past-the-end of the text to search.  
*re* [IN] The regular expression to search for.  
*submatches* [IN] A list of subexpressions to return for each regular expression match within the text.  
*m* [IN] Policy flags for match rules.

#### Todo

Implement this function.  
Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

5.680.2.4 `template<typename _Bi_iter , typename _Ch_type =  
typename iterator_traits<_Bi_iter>::value_type, typename  
_Rx_traits = regex_traits<_Ch_type>> template<std::size_t  
_Nm> std::regex_token_iterator< _Bi_iter, _Ch_type,  
_Rx_traits >::regex_token_iterator ( _Bi_iter __a,  
_Bi_iter __b, const regex_type & __re, const int(&  
__submatches[_Nm], regex_constants::match_flag_type __m =  
regex_constants::match_default )`

Constructs a `regex_token_iterator`...

#### Parameters

*a* [IN] The start of the text to search.  
*b* [IN] One-past-the-end of the text to search.  
*re* [IN] The regular expression to search for.  
*submatches* [IN] A list of subexpressions to return for each regular expression match within the text.  
*m* [IN] Policy flags for match rules.

#### Todo

Implement this function.  
Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

5.680.2.5 `template<typename _Bi_iter , typename _Ch_type = typename  
iterator_traits<_Bi_iter>::value_type, typename _Rx_traits  
= regex_traits<_Ch_type>> std::regex_token_iterator<  
_Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator ( const  
regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs )`

Copy constructs a `regex_token_iterator`.

#### Parameters

*rhs* [IN] A `regex_token_iterator` to copy.

#### Todo

Implement this function.

### 5.680.3 Member Function Documentation

**5.680.3.1** `template<typename _Bi_iter , typename _Ch_type = typename  
iterator_traits<_Bi_iter>::value_type, typename _Rx_traits =  
regex_traits<_Ch_type>> bool std::regex_token_iterator< _Bi_iter,  
_Ch_type, _Rx_traits >::operator!=( const regex_token_iterator<  
_Bi_iter, _Ch_type, _Rx_traits > & __rhs )`

Compares a `regex_token_iterator` to another for inequality.

#### Todo

Implement this function.

**5.680.3.2** `template<typename _Bi_iter , typename _Ch_type =  
typename iterator_traits<_Bi_iter>::value_type, typename  
_Rx_traits = regex_traits<_Ch_type>> const value_type&  
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits  
>::operator* ( )`

Dereferences a `regex_token_iterator`.

#### Todo

Implement this function.

**5.680.3.3** `template<typename _Bi_iter , typename _Ch_type =  
typename iterator_traits<_Bi_iter>::value_type, typename  
_Rx_traits = regex_traits<_Ch_type>> regex_token_iterator  
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits  
>::operator++ ( int )`



Postincrements a `regex_token_iterator`.

**Todo**

Implement this function.

**5.680.3.4** `template<typename _Bi_iter, typename _Ch_type =  
typename iterator_traits<_Bi_iter>::value_type, typename  
_Rx_traits = regex_traits<_Ch_type>> regex_token_iterator&  
std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits  
>::operator++ ( )`

Increments a `regex_token_iterator`.

**Todo**

Implement this function.

**5.680.3.5** `template<typename _Bi_iter, typename _Ch_type =  
typename iterator_traits<_Bi_iter>::value_type, typename  
_Rx_traits = regex_traits<_Ch_type>> const value_type*  
std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits  
>::operator-> ( )`

Selects a `regex_token_iterator` member.

**Todo**

Implement this function.

**5.680.3.6** `template<typename _Bi_iter, typename _Ch_type =  
typename iterator_traits<_Bi_iter>::value_type, typename  
_Rx_traits = regex_traits<_Ch_type>> regex_token_iterator&  
std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits  
>::operator= ( const regex_token_iterator<_Bi_iter, _Ch_type,  
_Rx_traits> & __rhs )`

Assigns a `regex_token_iterator` to another.

**Parameters**

*rhs* [IN] A `regex_token_iterator` to copy.

**Todo**

Implement this function.

```
5.680.3.7 template<typename _Bi_iter , typename _Ch_type = typename
iterator_traits<_Bi_iter>::value_type, typename _Rx_traits =
regex_traits<_Ch_type>> bool std::regex_token_iterator< _Bi_iter,
_Ch_type, _Rx_traits >::operator==(const regex_token_iterator<
_Bi_iter, _Ch_type, _Rx_traits > & __rhs)
```

Compares a `regex_token_iterator` to another for equality.

**Todo**

Implement this function.

The documentation for this class was generated from the following file:

- [regex.h](#)

**5.681 `std::regex_traits<_Ch_type>` Struct Template Reference**

Describes aspects of a regular expression.

**Public Types**

- typedef `std::ctype_base::mask` **char\_class\_type**
- typedef `_Ch_type` **char\_type**
- typedef [std::locale](#) **locale\_type**
- typedef [std::basic\\_string](#)< `char_type` > **string\_type**

**Public Member Functions**

- [regex\\_traits](#) ()
- [locale\\_type](#) `getloc` () const
- [locale\\_type](#) `imbue` ([locale\\_type](#) \_\_loc)
- bool [isctype](#) (`_Ch_type` \_\_c, `char_class_type` \_\_f) const

- template<typename \_Fwd\_iter >  
char\_class\_type [lookup\\_classname](#) (\_Fwd\_iter \_\_first, \_Fwd\_iter \_\_last, bool \_\_  
\_icase=false) const
- template<typename \_Fwd\_iter >  
[string\\_type](#) [lookup\\_collatename](#) (\_Fwd\_iter \_\_first, \_Fwd\_iter \_\_last) const
- template<typename \_Fwd\_iter >  
[string\\_type](#) [transform](#) (\_Fwd\_iter \_\_first, \_Fwd\_iter \_\_last) const
- template<typename \_Fwd\_iter >  
[string\\_type](#) [transform\\_primary](#) (\_Fwd\_iter \_\_first, \_Fwd\_iter \_\_last) const
- char\_type [translate](#) (char\_type \_\_c) const
- char\_type [translate\\_nocase](#) (char\_type \_\_c) const
- int [value](#) (\_Ch\_type \_\_ch, int \_\_radix) const

### Static Public Member Functions

- static std::size\_t [length](#) (const char\_type \*\_\_p)

### Protected Attributes

- [locale\\_type](#) [\\_M\\_locale](#)

#### 5.681.1 Detailed Description

**template<typename \_Ch\_type> struct std::regex\_traits< \_Ch\_type >**

Describes aspects of a regular expression. A regular expression traits class that satisfies the requirements of section [28.7].

The class regex is parameterized around a set of related types and functions used to complete the definition of its semantics. This class satisfies the requirements of such a traits class.

Definition at line 53 of file regex.h.

#### 5.681.2 Constructor & Destructor Documentation

**5.681.2.1 template<typename \_Ch\_type > std::regex\_traits< \_Ch\_type >::regex\_traits ( ) [inline]**

Constructs a default traits object.

Definition at line 65 of file regex.h.

### 5.681.3 Member Function Documentation

**5.681.3.1** `template<typename _Ch_type > locale_type std::regex_traits< _Ch_type >::getloc ( ) const [inline]`

Gets a copy of the current locale in use by the [regex\\_traits](#) object.

Definition at line 274 of file regex.h.

**5.681.3.2** `template<typename _Ch_type > locale_type std::regex_traits< _Ch_type >::imbue ( locale_type __loc ) [inline]`

Imbues the [regex\\_traits](#) object with a copy of a new locale.

#### Parameters

*loc* A locale.

#### Returns

a copy of the previous locale in use by the [regex\\_traits](#) object.

#### Note

Calling imbue with a different locale than the one currently in use invalidates all cached data held by \*this.

Definition at line 263 of file regex.h.

**5.681.3.3** `template<typename _Ch_type > static std::size_t std::regex_traits< _Ch_type >::length ( const char_type * __p ) [inline, static]`

Gives the length of a C-style string starting at \_\_p.

#### Parameters

*\_\_p* a pointer to the start of a character sequence.

#### Returns

the number of characters between \*\_\_p and the first default-initialized value of type char\_type. In other words, uses the C-string algorithm for determining the length of a sequence of characters.

Definition at line 79 of file `regex.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::length()`.

**5.681.3.4** `template<typename _Ch_type> template<typename _Fwd_iter>  
char_class_type std::regex_traits<_Ch_type>::lookup_classname(  
_Fwd_iter __first, _Fwd_iter __last, bool __icase = false) const  
[inline]`

Maps one or more characters to a named character classification.

#### Parameters

- first* beginning of the character sequence.
- last* one-past-the-end of the character sequence.
- icase* ignores the case of the classification name.

#### Returns

an unspecified value that represents the character classification named by the character sequence designated by the iterator range `[first, last)`. If `icase` is true, the returned mask identifies the classification regardless of the case of the characters to be matched (for example, `[[:lower:]]` is the same as `[[:alpha:]]`), otherwise a case-dependant classification is returned. The value returned shall be independent of the case of the characters in the character sequence. If the name is not recognized then returns a value that compares equal to 0.

At least the following names (or their wide-character equivalent) are supported.

- `d`
- `w`
- `s`
- `alnum`
- `alpha`
- `blank`
- `cntrl`
- `digit`
- `graph`

- lower
- print
- punct
- space
- upper
- xdigit

#### Todo

Implement this function.

Definition at line 219 of file `regex.h`.

Referenced by `std::regex_traits<_Ch_type>::isctype()`.

**5.681.3.5** `template<typename _Ch_type> template<typename _Fwd_iter>  
string_type std::regex_traits<_Ch_type>::lookup_collatename (  
_Fwd_iter __first, _Fwd_iter __last ) const [inline]`

Gets a collation element by name.

#### Parameters

*first* beginning of the collation element name.

*last* one-past-the-end of the collation element name.

#### Returns

a sequence of one or more characters that represents the collating element consisting of the character sequence designated by the iterator range `[first, last)`. Returns an empty string if the character sequence is not a valid collating element.

#### Todo

Implement this function.

Definition at line 176 of file `regex.h`.

**5.681.3.6** `template<typename _Ch_type> template<typename _Fwd_iter>  
string_type std::regex_traits<_Ch_type>::transform ( _Fwd_iter  
__first, _Fwd_iter __last ) const [inline]`

Gets a sort key for a character sequence.

#### Parameters

*first* beginning of the character sequence.

*last* one-past-the-end of the character sequence.

Returns a sort key for the character sequence designated by the iterator range [F1, F2) such that if the character sequence [G1, G2) sorts before the character sequence [H1, H2) then `v.transform(G1, G2) < v.transform(H1, H2)`.

What this really does is provide a more efficient way to compare a string to multiple other strings in locales with fancy collation rules and equivalence classes.

#### Returns

a locale-specific sort key equivalent to the input range.

#### Exceptions

[\*std::bad\\_cast\*](#) if the current locale does not have a collate facet.

Definition at line 132 of file `regex.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::data()`, `std::basic_string<_CharT, _Traits, _Alloc>::size()`, `std::collate<_CharT>::transform()`, and `std::use_facet()`.

**5.681.3.7** `template<typename _Ch_type> template<typename _Fwd_iter>  
string_type std::regex_traits<_Ch_type>::transform_primary ( _Fwd_iter  
__first, _Fwd_iter __last ) const [inline]`

Gets a sort key for a character sequence, independant of case.

#### Parameters

*first* beginning of the character sequence.

*last* one-past-the-end of the character sequence.

Effects: if `typeid(use_facet<collate<_Ch_type>>) == typeid(collate_byname<_Ch_type>)` and the form of the sort key returned by `collate_byname<_Ch_type>::transform(first, last)` is known and can be converted into a primary sort key then returns that key, otherwise returns an empty string.

#### Todo

Implement this function.

Definition at line 158 of file `regex.h`.

**5.681.3.8** `template<typename _Ch_type> char_type std::regex_traits<_Ch_type>::translate ( char_type __c ) const [inline]`

Performs the identity translation.

#### Parameters

*c* A character to the locale-specific character set.

#### Returns

*c*.

Definition at line 90 of file `regex.h`.

**5.681.3.9** `template<typename _Ch_type> char_type std::regex_traits<_Ch_type>::translate_nocase ( char_type __c ) const [inline]`

Translates a character into a case-insensitive equivalent.

#### Parameters

*c* A character to the locale-specific character set.

#### Returns

the locale-specific lower-case equivalent of *c*.

#### Exceptions

[\*std::bad\\_cast\*](#) if the imbued locale does not support the ctype facet.



Definition at line 103 of file regex.h.

References [std::tolower\(\)](#).

The documentation for this struct was generated from the following file:

- [regex.h](#)

## **5.682 std::remove\_all\_extents< \_Tp > Struct Template Reference**

[remove\\_all\\_extents](#)

### **Public Types**

- `typedef _Tp type`

#### **5.682.1 Detailed Description**

**template<typename \_Tp> struct std::remove\_all\_extents< \_Tp >**

[remove\\_all\\_extents](#)

Definition at line 476 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## **5.683 std::remove\_const< \_Tp > Struct Template Reference**

[remove\\_const](#)

### **Public Types**

- `typedef _Tp type`

#### **5.683.1 Detailed Description**

**template<typename \_Tp> struct std::remove\_const< \_Tp >**

[remove\\_const](#)

Definition at line 417 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.684 `std::remove_cv< _Tp >` Struct Template Reference

[remove\\_cv](#)

### Public Types

- typedef [remove\\_const](#)< typename [remove\\_volatile](#)< \_Tp >::type >::type **type**

#### 5.684.1 Detailed Description

`template<typename _Tp> struct std::remove_cv< _Tp >`

[remove\\_cv](#)

Definition at line 435 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.685 `std::remove_extent< _Tp >` Struct Template Reference

[remove\\_extent](#)

### Public Types

- typedef `_Tp` **type**

#### 5.685.1 Detailed Description

`template<typename _Tp> struct std::remove_extent< _Tp >`

[remove\\_extent](#)

Definition at line 463 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.686 `std::remove_pointer<_Tp>` Struct Template Reference

[remove\\_pointer](#)

Inherits `std::__remove_pointer_helper<_Tp, remove_cv<_Tp>::type>`.

### Public Types

- `typedef _Tp type`

#### 5.686.1 Detailed Description

`template<typename _Tp> struct std::remove_pointer<_Tp>`

[remove\\_pointer](#)

Definition at line 499 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.687 `std::remove_reference<_Tp>` Struct Template Reference

[remove\\_reference](#)

### Public Types

- `typedef _Tp type`

#### 5.687.1 Detailed Description

`template<typename _Tp> struct std::remove_reference<_Tp>`

[remove\\_reference](#)

Definition at line 544 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 5.688 `std::remove_volatile<_Tp>` Struct Template Reference

[remove\\_volatile](#)

**Public Types**

- `typedef _Tp type`

**5.688.1 Detailed Description**

`template<typename _Tp> struct std::remove_volatile< _Tp >`

[remove\\_volatile](#)

Definition at line 426 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

**5.689 std::reverse\_iterator< \_Iterator > Class Template Reference**

Inheritance diagram for `std::reverse_iterator< _Iterator >`:

**Public Types**

- `typedef __traits_type::difference_type` [difference\\_type](#)
- `typedef iterator_traits< _Iterator >::iterator_category` [iterator\\_category](#)
- `typedef _Iterator` **iterator\_type**
- `typedef __traits_type::pointer` [pointer](#)
- `typedef __traits_type::reference` [reference](#)
- `typedef iterator_traits< _Iterator >::value_type` [value\\_type](#)

**Public Member Functions**

- [reverse\\_iterator](#) ()
- [reverse\\_iterator](#) (iterator\_type \_\_x)
- `template<typename _Iter >`  
[reverse\\_iterator](#) (const [reverse\\_iterator](#)< \_Iter > &\_\_x)
- [reverse\\_iterator](#) (const [reverse\\_iterator](#) &\_\_x)
- iterator\_type [base](#) () const
- [reference operator\\*](#) () const

- `reverse_iterator operator+ (difference_type __n) const`
- `reverse_iterator & operator++ ()`
- `reverse_iterator operator++ (int)`
- `reverse_iterator & operator+= (difference_type __n)`
- `reverse_iterator operator- (difference_type __n) const`
- `reverse_iterator & operator-- ()`
- `reverse_iterator operator-- (int)`
- `reverse_iterator & operator-= (difference_type __n)`
- `pointer operator-> () const`
- `reference operator[] (difference_type __n) const`

### Protected Types

- `typedef iterator_traits<_Iterator> __traits_type`

### Protected Attributes

- `_Iterator current`

#### 5.689.1 Detailed Description

`template<typename _Iterator> class std::reverse_iterator<_Iterator>`

Bidirectional and random access iterators have corresponding reverse iterator adaptors that iterate through the data structure in the opposite direction. They have the same signatures as the corresponding iterators. The fundamental relation between a reverse iterator and its corresponding iterator `i` is established by the identity:

```
&*(reverse_iterator(i)) == &*(i - 1)
```

*This mapping is dictated by the fact that while there is always a pointer past the end of an array, there might not be a valid pointer before the beginning of an array.* [24.4.1]/1,2

Reverse iterators can be tricky and surprising at first. Their semantics make sense, however, and the trickiness is a side effect of the requirement that the iterators must be safe.

Definition at line 97 of file `stl_iterator.h`.

## 5.689.2 Member Typedef Documentation

**5.689.2.1** `template<typename _Iterator> typedef __traits_difference_type std::reverse_iterator< _Iterator >::difference_type`

Distance between iterators is represented as this type.

Reimplemented from [std::iterator< iterator\\_traits< \\_Iterator >::iterator\\_category, iterator\\_traits< \\_Iterator >::value\\_type, iterator\\_traits< \\_Iterator >::difference\\_type, iterator\\_traits< \\_Iterator >::pointer, iterator\\_traits< \\_Iterator >::reference >](#).

Definition at line 111 of file `stl_iterator.h`.

**5.689.2.2** `typedef iterator_traits< _Iterator >::iterator_category std::iterator< iterator_traits< _Iterator >::iterator_category, iterator_traits< _Iterator >::value_type, iterator_traits< _Iterator >::difference_type, iterator_traits< _Iterator >::pointer, iterator_traits< _Iterator >::reference >::iterator_category [inherited]`

One of the [tag types](#).

Definition at line 122 of file `stl_iterator_base_types.h`.

**5.689.2.3** `template<typename _Iterator> typedef __traits_type::pointer std::reverse_iterator< _Iterator >::pointer`

This type represents a pointer-to-value\_type.

Reimplemented from [std::iterator< iterator\\_traits< \\_Iterator >::iterator\\_category, iterator\\_traits< \\_Iterator >::value\\_type, iterator\\_traits< \\_Iterator >::difference\\_type, iterator\\_traits< \\_Iterator >::pointer, iterator\\_traits< \\_Iterator >::reference >](#).

Definition at line 112 of file `stl_iterator.h`.

**5.689.2.4** `template<typename _Iterator> typedef __traits_type::reference std::reverse_iterator< _Iterator >::reference`

This type represents a reference-to-value\_type.

Reimplemented from [std::iterator< iterator\\_traits< \\_Iterator >::iterator\\_category, iterator\\_traits< \\_Iterator >::value\\_type, iterator\\_traits< \\_Iterator >::difference\\_type, iterator\\_traits< \\_Iterator >::pointer, iterator\\_traits< \\_Iterator >::reference >](#).

Definition at line 113 of file stl\_iterator.h.

```
5.689.2.5 typedef iterator_traits< _Iterator >::value_type std::iterator<
 iterator_traits< _Iterator >::iterator_category , iterator_traits<
 _Iterator >::value_type , iterator_traits< _Iterator
 >::difference_type , iterator_traits< _Iterator >::pointer
 , iterator_traits< _Iterator >::reference >::value_type
 [inherited]
```

The type "pointed to" by the iterator.

Definition at line 124 of file stl\_iterator\_base\_types.h.

### 5.689.3 Constructor & Destructor Documentation

```
5.689.3.1 template<typename _Iterator> std::reverse_iterator< _Iterator
 >::reverse_iterator () [inline]
```

The default constructor default-initializes member `current`. If it is a pointer, that means it is zero-initialized.

Definition at line 121 of file stl\_iterator.h.

Referenced by `std::reverse_iterator< _Iterator >::operator+()`, and `std::reverse_iterator< _Iterator >::operator-()`.

```
5.689.3.2 template<typename _Iterator> std::reverse_iterator< _Iterator
 >::reverse_iterator (iterator_type __x) [inline, explicit]
```

This iterator will move in the opposite direction that `x` does.

Definition at line 127 of file stl\_iterator.h.

```
5.689.3.3 template<typename _Iterator> std::reverse_iterator< _Iterator
 >::reverse_iterator (const reverse_iterator< _Iterator > & __x)
 [inline]
```

The copy constructor is normal.

Definition at line 132 of file stl\_iterator.h.

**5.689.3.4** `template<typename _Iterator> template<typename _Iter >  
std::reverse_iterator< _Iterator >::reverse_iterator ( const  
reverse_iterator< _Iter > & __x ) [inline]`

A [reverse\\_iterator](#) across other types can be copied in the normal fashion.

Definition at line 140 of file `stl_iterator.h`.

#### 5.689.4 Member Function Documentation

**5.689.4.1** `template<typename _Iterator> iterator_type std::reverse_iterator<  
_Iterator >::base ( ) const [inline]`

##### Returns

`current`, the iterator used for underlying work.

Definition at line 147 of file `stl_iterator.h`.

Referenced by `std::operator==( )`.

**5.689.4.2** `template<typename _Iterator> reference std::reverse_iterator<  
_Iterator >::operator* ( ) const [inline]`

##### Returns

TODO

##### Todo

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

Definition at line 156 of file `stl_iterator.h`.

Referenced by `std::reverse_iterator< _Iterator >::operator->( )`.

**5.689.4.3** `template<typename _Iterator> reverse_iterator  
std::reverse_iterator< _Iterator >::operator+ ( difference_type __n  
) const [inline]`



**Returns**

TODO

**Todo**

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

Definition at line 227 of file stl\_iterator.h.

References std::reverse\_iterator< \_Iterator >::reverse\_iterator().

**5.689.4.4** `template<typename _Iterator> reverse_iterator&  
std::reverse_iterator< _Iterator >::operator++ ( ) [inline]`

**Returns**

TODO

**Todo**

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

Definition at line 177 of file stl\_iterator.h.

**5.689.4.5** `template<typename _Iterator> reverse_iterator  
std::reverse_iterator< _Iterator >::operator++ ( int ) [inline]`

**Returns**

TODO

**Todo**

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

Definition at line 189 of file stl\_iterator.h.

**5.689.4.6** `template<typename _Iterator> reverse_iterator&  
std::reverse_iterator< _Iterator >::operator+=( difference_type  
__n ) [inline]`

#### Returns

TODO

#### Todo

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

Definition at line 236 of file stl\_iterator.h.

**5.689.4.7** `template<typename _Iterator> reverse_iterator  
std::reverse_iterator< _Iterator >::operator- ( difference_type __n  
) const [inline]`

#### Returns

TODO

#### Todo

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

Definition at line 248 of file stl\_iterator.h.

References `std::reverse_iterator< _Iterator >::reverse_iterator()`.

**5.689.4.8** `template<typename _Iterator> reverse_iterator  
std::reverse_iterator< _Iterator >::operator-- ( int ) [inline]`

#### Returns

TODO

#### Todo

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

Definition at line 214 of file stl\_iterator.h.

**5.689.4.9** `template<typename _Iterator> reverse_iterator&  
std::reverse_iterator< _Iterator >::operator-- ( ) [inline]`

**Returns**

TODO

**Todo**

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

Definition at line 202 of file stl\_iterator.h.

**5.689.4.10** `template<typename _Iterator> reverse_iterator&  
std::reverse_iterator< _Iterator >::operator-= ( difference_type  
__n ) [inline]`

**Returns**

TODO

**Todo**

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

Definition at line 257 of file stl\_iterator.h.

**5.689.4.11** `template<typename _Iterator> pointer std::reverse_iterator<  
_Iterator >::operator-> ( ) const [inline]`

**Returns**

TODO

**Todo**

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

Definition at line 168 of file stl\_iterator.h.

References std::reverse\_iterator< \_Iterator >::operator\*().

**5.689.4.12** `template<typename _Iterator> reference std::reverse_iterator<_Iterator>::operator[] ( difference_type __n ) const [inline]`

#### Returns

TODO

#### Todo

Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

Definition at line 269 of file stl\_iterator.h.

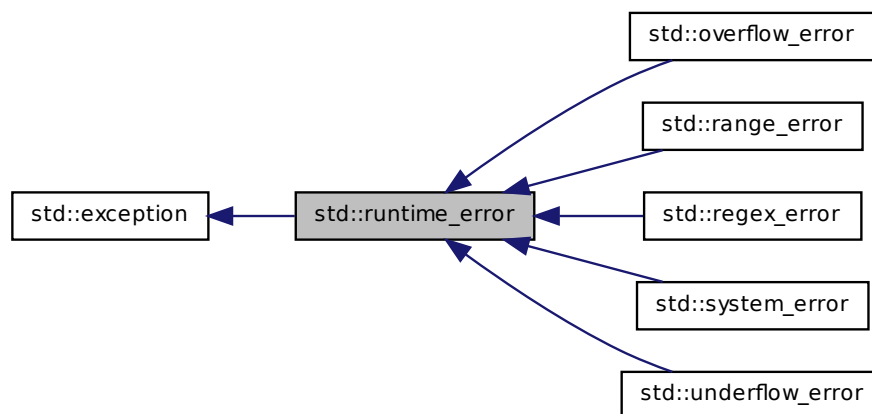
The documentation for this class was generated from the following file:

- [stl\\_iterator.h](#)

## 5.690 std::runtime\_error Class Reference

One of two subclasses of exception.

Inheritance diagram for std::runtime\_error:



## Public Member Functions

- `runtime_error` (const `string` &\_\_arg)
- virtual const char \* `what` () const throw ()

### 5.690.1 Detailed Description

One of two subclasses of exception. Runtime errors represent problems outside the scope of a program; they cannot be easily predicted and can generally only be caught as the program executes.

Definition at line 110 of file `stdexcept`.

### 5.690.2 Constructor & Destructor Documentation

#### 5.690.2.1 `std::runtime_error::runtime_error ( const string & __arg )` [`explicit`]

Takes a character string describing the error.

### 5.690.3 Member Function Documentation

#### 5.690.3.1 `virtual const char* std::runtime_error::what ( ) const throw ()` [`virtual`]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from `std::exception`.

The documentation for this class was generated from the following file:

- `stdexcept`

## 5.691 `std::seed_seq` Class Reference

The `seed_seq` class generates sequences of seeds for random number generators.

## Public Types

- typedef uint\_least32\_t `result_type`

## Public Member Functions

- [seed\\_seq](#) ()
- `template<typename _IntType >`  
`seed_seq` ([std::initializer\\_list](#)< \_IntType > il)
- `template<typename _InputIterator >`  
`seed_seq` (\_InputIterator \_\_begin, \_InputIterator \_\_end)
- `template<typename _RandomAccessIterator >`  
`void generate` (\_RandomAccessIterator \_\_begin, \_RandomAccessIterator \_\_-  
end)
- `template<typename OutputIterator >`  
`void param` (OutputIterator \_\_dest) const
- `size_t size` () const

### 5.691.1 Detailed Description

The [seed\\_seq](#) class generates sequences of seeds for random number generators.

Definition at line 5349 of file random.h.

### 5.691.2 Member Typedef Documentation

#### 5.691.2.1 `typedef uint_least32_t std::seed_seq::result_type`

The type of the seed vales.

Definition at line 5354 of file random.h.

### 5.691.3 Constructor & Destructor Documentation

#### 5.691.3.1 `std::seed_seq::seed_seq ( ) [inline]`

Default constructor.

Definition at line 5357 of file random.h.

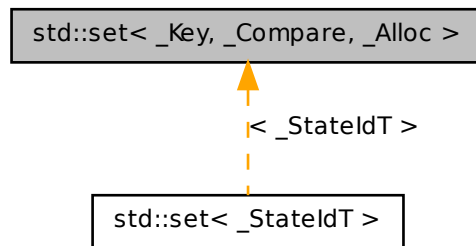
The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

**5.692 `std::set< _Key, _Compare, _Alloc >` Class Template Reference**

A standard container made up of unique keys, which can be retrieved in logarithmic time.

Inheritance diagram for `std::set< _Key, _Compare, _Alloc >`:

**Public Types**

- `typedef _Key` [key\\_type](#)
- `typedef _Key` [value\\_type](#)
- `typedef _Compare` [key\\_compare](#)
- `typedef _Compare` [value\\_compare](#)
- `typedef _Alloc` [allocator\\_type](#)
  
- `typedef _Key_alloc_type::pointer` [pointer](#)
- `typedef _Key_alloc_type::const_pointer` [const\\_pointer](#)
- `typedef _Key_alloc_type::reference` [reference](#)
- `typedef _Key_alloc_type::const_reference` [const\\_reference](#)
- `typedef _Rep_type::const_iterator` [iterator](#)
- `typedef _Rep_type::const_iterator` [const\\_iterator](#)
- `typedef _Rep_type::const_reverse_iterator` [reverse\\_iterator](#)
- `typedef _Rep_type::const_reverse_iterator` [const\\_reverse\\_iterator](#)
- `typedef _Rep_type::size_type` [size\\_type](#)
- `typedef _Rep_type::difference_type` [difference\\_type](#)

**Public Member Functions**

- [set](#) ()

- [set](#) (const [\\_Compare](#) &\_\_comp, const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
  - [template](#)<typename [\\_InputIterator](#) >  
[set](#) ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last, const [\\_Compare](#) &\_\_comp,  
const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
  - [set](#) (const [set](#) &\_\_x)
  - [template](#)<typename [\\_InputIterator](#) >  
[set](#) ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last)
  - [set](#) ([set](#) &&\_\_x)
  - [set](#) ([initializer\\_list](#)< [value\\_type](#) > \_\_l, const [\\_Compare](#) &\_\_comp=[\\_Compare](#)(),  
const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
  - [iterator begin](#) () const
  - [iterator cbegin](#) () const
  - [iterator cend](#) () const
  - void [clear](#) ()
  - [size\\_type count](#) (const [key\\_type](#) &\_\_x) const
  - [reverse\\_iterator crbegin](#) () const
  - [reverse\\_iterator crend](#) () const
  - bool [empty](#) () const
  - [iterator end](#) () const
  - [size\\_type erase](#) (const [key\\_type](#) &\_\_x)
  - [iterator erase](#) (const [iterator](#) \_\_position)
  - [iterator erase](#) (const [iterator](#) \_\_first, const [iterator](#) \_\_last)
  - [allocator\\_type get\\_allocator](#) () const
  - [iterator insert](#) (const [iterator](#) \_\_position, [value\\_type](#) &&\_\_x)
  - [std::pair](#)< [iterator](#), bool > [insert](#) ([value\\_type](#) &&\_\_x)
  - void [insert](#) ([initializer\\_list](#)< [value\\_type](#) > \_\_l)
  - [iterator insert](#) (const [iterator](#) \_\_position, const [value\\_type](#) &\_\_x)
  - [std::pair](#)< [iterator](#), bool > [insert](#) (const [value\\_type](#) &\_\_x)
  - [template](#)<typename [\\_InputIterator](#) >  
void [insert](#) ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last)
  - [key\\_compare key\\_comp](#) () const
  - [size\\_type max\\_size](#) () const
  - [set](#) & [operator=](#) (const [set](#) &\_\_x)
  - [set](#) & [operator=](#) ([set](#) &&\_\_x)
  - [set](#) & [operator=](#) ([initializer\\_list](#)< [value\\_type](#) > \_\_l)
  - [reverse\\_iterator rbegin](#) () const
  - [reverse\\_iterator rend](#) () const
  - [size\\_type size](#) () const
  - void [swap](#) ([set](#) &\_\_x)
  - [value\\_compare value\\_comp](#) () const
- 
- [iterator find](#) (const [key\\_type](#) &\_\_x)
  - [const\\_iterator find](#) (const [key\\_type](#) &\_\_x) const



- `iterator lower_bound` (const `key_type` &\_\_x)
- `const_iterator lower_bound` (const `key_type` &\_\_x) const
- `iterator upper_bound` (const `key_type` &\_\_x)
- `const_iterator upper_bound` (const `key_type` &\_\_x) const
- `std::pair< iterator, iterator > equal_range` (const `key_type` &\_\_x)
- `std::pair< const_iterator, const_iterator > equal_range` (const `key_type` &\_\_x) const

### Friends

- `template<typename _K1, typename _C1, typename _A1 >`  
`bool operator<` (const `set< _K1, _C1, _A1 >` &, const `set< _K1, _C1, _A1 >` &)
- `template<typename _K1, typename _C1, typename _A1 >`  
`bool operator==` (const `set< _K1, _C1, _A1 >` &, const `set< _K1, _C1, _A1 >` &)

#### 5.692.1 Detailed Description

`template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> class std::set<_Key, _Compare, _Alloc>`

A standard container made up of unique keys, which can be retrieved in logarithmic time. Meets the requirements of a `container`, a `reversible container`, and an `associative container` (using unique keys).

Sets support bidirectional iterators.

### Parameters

**Key** Type of key objects.

**Compare** Comparison function object type, defaults to `less<Key>`.

**Alloc** Allocator type, defaults to `allocator<Key>`.

The private tree data is declared exactly the same way for set and multiset; the distinction is made entirely in how the tree functions are called (`*_unique` versus `*_equal`, same as the standard).

Definition at line 89 of file `stl_set.h`.

### 5.692.2 Member Typedef Documentation

**5.692.2.1** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> typedef _Alloc std::set<  
_Key, _Compare, _Alloc >::allocator_type`

Public typedefs.

Definition at line 106 of file stl\_set.h.

**5.692.2.2** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> typedef  
_Rep_type::const_iterator std::set< _Key, _Compare, _Alloc  
>::const_iterator`

Iterator-related typedefs.

Definition at line 127 of file stl\_set.h.

**5.692.2.3** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> typedef  
_Key_alloc_type::const_pointer std::set< _Key, _Compare, _Alloc  
>::const_pointer`

Iterator-related typedefs.

Definition at line 120 of file stl\_set.h.

**5.692.2.4** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> typedef  
_Key_alloc_type::const_reference std::set< _Key, _Compare, _Alloc  
>::const_reference`

Iterator-related typedefs.

Definition at line 122 of file stl\_set.h.

**5.692.2.5** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> typedef  
_Rep_type::const_reverse_iterator std::set<_Key, _Compare, _Alloc  
>::const_reverse_iterator`

Iterator-related typedefs.

Definition at line 129 of file stl\_set.h.

**5.692.2.6** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> typedef  
_Rep_type::difference_type std::set<_Key, _Compare, _Alloc  
>::difference_type`

Iterator-related typedefs.

Definition at line 131 of file stl\_set.h.

**5.692.2.7** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> typedef  
_Rep_type::const_iterator std::set<_Key, _Compare, _Alloc  
>::iterator`

Iterator-related typedefs.

Definition at line 126 of file stl\_set.h.

**5.692.2.8** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> typedef _Compare  
std::set<_Key, _Compare, _Alloc>::key_compare`

Public typedefs.

Definition at line 104 of file stl\_set.h.

**5.692.2.9** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> typedef _Key std::set<  
_Key, _Compare, _Alloc>::key_type`

Public typedefs.

Definition at line 102 of file `stl_set.h`.

**5.692.2.10** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> typedef  
_Key_alloc_type::pointer std::set<_Key, _Compare, _Alloc  
>::pointer`

Iterator-related typedefs.

Definition at line 119 of file `stl_set.h`.

**5.692.2.11** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> typedef  
_Key_alloc_type::reference std::set<_Key, _Compare, _Alloc  
>::reference`

Iterator-related typedefs.

Definition at line 121 of file `stl_set.h`.

**5.692.2.12** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> typedef  
_Rep_type::const_reverse_iterator std::set<_Key, _Compare,  
_Alloc>::reverse_iterator`

Iterator-related typedefs.

Definition at line 128 of file `stl_set.h`.

**5.692.2.13** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> typedef  
_Rep_type::size_type std::set<_Key, _Compare, _Alloc  
>::size_type`

Iterator-related typedefs.

Definition at line 130 of file stl\_set.h.

**5.692.2.14** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> typedef _Compare  
std::set<_Key, _Compare, _Alloc>::value_compare`

Public typedefs.

Definition at line 105 of file stl\_set.h.

**5.692.2.15** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> typedef _Key std::set<  
_Key, _Compare, _Alloc>::value_type`

Public typedefs.

Definition at line 103 of file stl\_set.h.

### 5.692.3 Constructor & Destructor Documentation

**5.692.3.1** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> std::set<_Key,  
_Compare, _Alloc>::set( ) [inline]`

Default constructor creates no elements.

Definition at line 138 of file stl\_set.h.

**5.692.3.2** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> std::set<_Key,  
_Compare, _Alloc>::set ( const _Compare & __comp, const  
allocator_type & __a = allocator_type () ) [inline,  
explicit]`

Creates a set with no elements.

#### Parameters

*comp* Comparator to use.

*a* An allocator object.

Definition at line 147 of file stl\_set.h.

**5.692.3.3** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> template<typename  
_InputIterator> std::set<_Key, _Compare, _Alloc>::set (  
_InputIterator __first, _InputIterator __last ) [inline]`

Builds a set from a range.

#### Parameters

*first* An input iterator.

*last* An input iterator.

Create a set consisting of copies of the elements from [first,last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(first,last)).

Definition at line 161 of file stl\_set.h.

**5.692.3.4** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> template<typename  
_InputIterator> std::set<_Key, _Compare, _Alloc>::set (  
_InputIterator __first, _InputIterator __last, const _Compare  
& __comp, const allocator_type & __a = allocator_type () )  
[inline]`

Builds a set from a range.

**Parameters**

- first* An input iterator.  
*last* An input iterator.  
*comp* A comparison functor.  
*a* An allocator object.

Create a set consisting of copies of the elements from [first,last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(first,last)).

Definition at line 177 of file `stl_set.h`.

**5.692.3.5** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> std::set<_Key,  
_Compare, _Alloc>::set ( const set<_Key, _Compare, _Alloc> &  
__x ) [inline]`

Set copy constructor.

**Parameters**

- x* A set of identical element and allocator types.

The newly-created set uses a copy of the allocation object used by *x*.

Definition at line 190 of file `stl_set.h`.

**5.692.3.6** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> std::set<_Key,  
_Compare, _Alloc>::set ( set<_Key, _Compare, _Alloc> && __x  
) [inline]`

Set move constructor

**Parameters**

- x* A set of identical element and allocator types.

The newly-created set contains the exact contents of *x*. The contents of *x* are a valid, but unspecified set.

Definition at line 201 of file `stl_set.h`.

**5.692.3.7** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> std::set<_Key,  
_Compare, _Alloc>::set ( initializer_list< value_type > __l, const  
_Compare & __comp = _Compare(), const allocator_type & __a =  
allocator_type() ) [inline]`

Builds a set from an [initializer\\_list](#).

#### Parameters

- l* An [initializer\\_list](#).
- comp* A comparison functor.
- a* An allocator object.

Create a set consisting of copies of the elements in the list. This is linear in N if the list is already sorted, and NlogN otherwise (where N is *l.size()*).

Definition at line 214 of file `stl_set.h`.

### 5.692.4 Member Function Documentation

**5.692.4.1** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> iterator std::set<_Key,  
_Compare, _Alloc>::begin ( ) const [inline]`

Returns a read-only (constant) iterator that points to the first element in the set. Iteration is done in ascending order according to the keys.

Definition at line 294 of file `stl_set.h`.

**5.692.4.2** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> iterator std::set<_Key,  
_Compare, _Alloc>::cbegin ( ) const [inline]`

Returns a read-only (constant) iterator that points to the first element in the set. Iteration is done in ascending order according to the keys.

Definition at line 331 of file `stl_set.h`.



**5.692.4.3** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> iterator std::set<_Key,  
_Compare, _Alloc>::end ( ) const [inline]`

Returns a read-only (constant) iterator that points one past the last element in the set. Iteration is done in ascending order according to the keys.

Definition at line 340 of file stl\_set.h.

**5.692.4.4** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> void std::set<_Key,  
_Compare, _Alloc>::clear ( ) [inline]`

Erases all elements in a set. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 572 of file stl\_set.h.

Referenced by `std::set<_StateIdT>::operator=()`.

**5.692.4.5** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> size_type std::set<  
_Key, _Compare, _Alloc>::count ( const key_type & __x ) const  
[inline]`

Finds the number of elements.

#### Parameters

*x* Element to located.

#### Returns

Number of elements with specified key.

This function only makes sense for multisets; for set the result will either be 0 (not present) or 1 (present).

Definition at line 586 of file stl\_set.h.

**5.692.4.6** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> reverse_iterator std::set<  
_Key, _Compare, _Alloc>::crbegin ( ) const [inline]`

Returns a read-only (constant) iterator that points to the last element in the set. Iteration is done in descending order according to the keys.

Definition at line 349 of file stl\_set.h.

**5.692.4.7** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> reverse_iterator std::set<  
_Key, _Compare, _Alloc>::crend ( ) const [inline]`

Returns a read-only (constant) reverse iterator that points to the last pair in the set. Iteration is done in descending order according to the keys.

Definition at line 358 of file stl\_set.h.

**5.692.4.8** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> bool std::set<_Key,  
_Compare, _Alloc>::empty ( ) const [inline]`

Returns true if the set is empty.

Definition at line 364 of file stl\_set.h.

**5.692.4.9** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> iterator std::set<_Key,  
_Compare, _Alloc>::end ( ) const [inline]`

Returns a read-only (constant) iterator that points one past the last element in the set. Iteration is done in ascending order according to the keys.

Definition at line 303 of file stl\_set.h.

**5.692.4.10** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> std::pair<iterator,  
iterator> std::set<_Key, _Compare, _Alloc>::equal_range ( const  
key_type & __x ) [inline]`

Finds a subsequence matching given key.

**Parameters**

*x* Key to be located.

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
 c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 666 of file `stl_set.h`.

```
5.692.4.11 template<typename _Key, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::pair<const_iterator, const_iterator> std::set<_Key,
_Compare, _Alloc>::equal_range (const key_type & __x) const
[inline]
```

Finds a subsequence matching given key.

**Parameters**

*x* Key to be located.

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
 c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 670 of file `stl_set.h`.

**5.692.4.12** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> iterator std::set<  
_Key, _Compare, _Alloc>::erase ( const_iterator __position )  
[inline]`

Erases an element from a set.

#### Parameters

*position* An iterator pointing to the element to be erased.

#### Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 497 of file `stl_set.h`.

**5.692.4.13** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> size_type std::set<_Key,  
_Compare, _Alloc>::erase ( const key_type & __x ) [inline]`

Erases elements according to the provided key.

#### Parameters

*x* Key of element to be erased.

#### Returns

The number of elements erased.

This function erases all the elements located by the given key from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 527 of file `stl_set.h`.

**5.692.4.14** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> iterator std::set<_Key,  
_Compare, _Alloc>::erase ( const_iterator __first, const_iterator  
__last ) [inline]`

Erases a [first,last) range of elements from a set.

#### Parameters

*first* Iterator pointing to the start of the range to be erased.

*last* Iterator pointing to the end of the range to be erased.

#### Returns

The iterator *last*.

This function erases a sequence of elements from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 546 of file stl\_set.h.

**5.692.4.15** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> iterator std::set<_Key,  
_Compare, _Alloc>::find ( const key_type & __x ) [inline]`

Tries to locate an element in a set.

#### Parameters

*x* Element to be located.

#### Returns

Iterator pointing to sought-after element, or [end\(\)](#) if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( [end\(\)](#) ) iterator.

Definition at line 604 of file stl\_set.h.

**5.692.4.16** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> const_iterator std::set<  
_Key, _Compare, _Alloc >::find ( const key_type & __x ) const  
[inline]`

Tries to locate an element in a set.

#### Parameters

*x* Element to be located.

#### Returns

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 608 of file `stl_set.h`.

**5.692.4.17** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> allocator_type std::set<  
_Key, _Compare, _Alloc >::get_allocator ( ) const [inline]`

Returns the allocator object with which the set was constructed.

Definition at line 285 of file `stl_set.h`.

**5.692.4.18** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> std::pair<iterator,  
bool> std::set<_Key, _Compare, _Alloc >::insert ( const  
value_type & __x ) [inline]`

Attempts to insert an element into the set.

#### Parameters

*x* Element to be inserted.

#### Returns

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to insert an element into the set. A set relies on unique keys and thus an element is only inserted if it is not already present in the set.

Insertion requires logarithmic time.

Definition at line 407 of file stl\_set.h.

Referenced by std::set< \_StateIdT >::operator=().

**5.692.4.19** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> iterator std::set< _Key,  
_Compare, _Alloc >::insert ( const_iterator __position, const  
value_type & __x ) [inline]`

Attempts to insert an element into the set.

#### Parameters

*position* An iterator that serves as a hint as to where the element should be inserted.

*x* Element to be inserted.

#### Returns

An iterator that points to the element with key of *x* (may or may not be the element passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument [insert\(\)](#) does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.htm>

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 444 of file stl\_set.h.

**5.692.4.20** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> template<typename  
_InputIterator > void std::set< _Key, _Compare, _Alloc >::insert (   
_InputIterator __first, _InputIterator __last ) [inline]`

A template function that attempts to insert a range of elements.

**Parameters**

*first* Iterator pointing to the start of the range to be inserted.

*last* Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 464 of file `stl_set.h`.

**5.692.4.21** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> void std::set<_Key,  
_Compare, _Alloc>::insert ( initializer_list<value_type> & __l )  
[inline]`

Attempts to insert a list of elements into the set.

**Parameters**

*list* A `std::initializer_list<value_type>` of elements to be inserted.

Complexity similar to that of the range constructor.

Definition at line 476 of file `stl_set.h`.

Referenced by `std::set<_StateIdT>::insert()`.

**5.692.4.22** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> const _Compare& std::set<  
_Key, _Compare, _Alloc>::key_comp ( ) const [inline]`

Returns the comparison object with which the set was constructed.

Definition at line 277 of file `stl_set.h`.

**5.692.4.23** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> iterator std::set<_Key,  
_Compare, _Alloc>::lower_bound ( const key_type & __x )  
[inline]`

Finds the beginning of a subsequence matching given key.



**Parameters**

*x* Key to be located.

**Returns**

Iterator pointing to first element equal to or greater than key, or `end()`.

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or `end()` if no such element exists.

Definition at line 625 of file `stl_set.h`.

**5.692.4.24** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> const_iterator std::set<  
_Key, _Compare, _Alloc>::lower_bound ( const key_type & __x )  
const [inline]`

Finds the beginning of a subsequence matching given key.

**Parameters**

*x* Key to be located.

**Returns**

Iterator pointing to first element equal to or greater than key, or `end()`.

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or `end()` if no such element exists.

Definition at line 629 of file `stl_set.h`.

**5.692.4.25** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> size_type std::set<  
_Key, _Compare, _Alloc>::max_size ( ) const [inline]`

Returns the maximum size of the set.

Definition at line 374 of file `stl_set.h`.

**5.692.4.26** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> set& std::set<_Key,  
_Compare, _Alloc>::operator= ( set<_Key, _Compare, _Alloc>  
&& __x ) [inline]`

Set move assignment operator.

#### Parameters

*x* A set of identical element and allocator types.

The contents of *x* are moved into this set (without copying). *x* is a valid, but unspecified set.

Definition at line 244 of file stl\_set.h.

**5.692.4.27** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> set& std::set<_Key,  
_Compare, _Alloc>::operator= ( const set<_Key, _Compare,  
_Alloc> & __x ) [inline]`

Set assignment operator.

#### Parameters

*x* A set of identical element and allocator types.

All the elements of *x* are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 229 of file stl\_set.h.

**5.692.4.28** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> set& std::set<_Key,  
_Compare, _Alloc>::operator= ( initializer_list<value_type> __l  
) [inline]`

Set list assignment operator.

#### Parameters

*l* An [initializer\\_list](#).

This function fills a set with copies of the elements in the initializer list *l*.

Note that the assignment completely changes the set and that the resulting set's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 265 of file stl\_set.h.

**5.692.4.29** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> reverse_iterator  
std::set<_Key, _Compare, _Alloc>::rbegin ( ) const [inline]`

Returns a read-only (constant) iterator that points to the last element in the set. Iteration is done in descending order according to the keys.

Definition at line 312 of file stl\_set.h.

**5.692.4.30** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> reverse_iterator  
std::set<_Key, _Compare, _Alloc>::rend ( ) const [inline]`

Returns a read-only (constant) reverse iterator that points to the last pair in the set. Iteration is done in descending order according to the keys.

Definition at line 321 of file stl\_set.h.

**5.692.4.31** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> size_type std::set<  
_Key, _Compare, _Alloc>::size ( ) const [inline]`

Returns the size of the set.

Definition at line 369 of file stl\_set.h.

**5.692.4.32** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> void std::set<_Key,  
_Compare, _Alloc>::swap ( set<_Key, _Compare, _Alloc> & __x  
) [inline]`

Swaps data with another set.

#### Parameters

*x* A set of the same element and allocator types.

This exchanges the elements between two sets in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(s1,s2)` will feed to this function.

Definition at line 389 of file `stl_set.h`.

Referenced by `std::set<_StateIdT>::operator=()`, and `std::swap()`.

**5.692.4.33** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> const_iterator std::set<  
_Key, _Compare, _Alloc>::upper_bound ( const key_type & __x )  
const [inline]`

Finds the end of a subsequence matching given key.

#### Parameters

*x* Key to be located.

#### Returns

Iterator pointing to the first element greater than key, or `end()`.

Definition at line 645 of file `stl_set.h`.

**5.692.4.34** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> iterator std::set<_Key,  
_Compare, _Alloc>::upper_bound ( const key_type & __x )  
[inline]`

Finds the end of a subsequence matching given key.

#### Parameters

*x* Key to be located.

#### Returns

Iterator pointing to the first element greater than key, or `end()`.

Definition at line 641 of file `stl_set.h`.

**5.692.4.35** `template<typename _Key, typename _Compare = std::less<_Key>,  
typename _Alloc = std::allocator<_Key>> value_compare std::set<  
_Key, _Compare, _Alloc>::value_comp ( ) const [inline]`

Returns the comparison object with which the set was constructed.

Definition at line 281 of file `stl_set.h`.

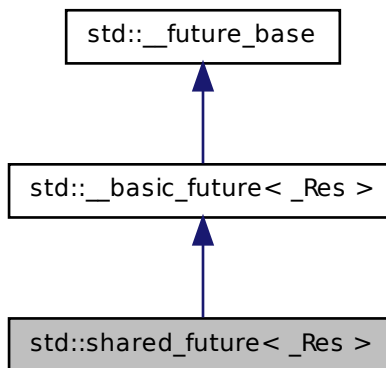
The documentation for this class was generated from the following file:

- [stl\\_set.h](#)

## 5.693 `std::shared_future<_Res>` Class Template Reference

Primary template for [shared\\_future](#).

Inheritance diagram for `std::shared_future<_Res>`:



### Public Member Functions

- [shared\\_future](#) (const [shared\\_future](#) &\_\_sf)
- [shared\\_future](#) ([shared\\_future](#) &&\_\_sf)
- [shared\\_future](#) ([future](#)<\_Res> &&\_\_uf)
- const \_Res & [get](#) ()

- `shared_future` & **operator=** (const `shared_future` &\_\_sf)
- `shared_future` & **operator=** (`shared_future` &&\_\_sf)
- bool **valid** () const
- void **wait** () const
- template<typename \_Rep, typename \_Period >  
bool **wait\_for** (const `chrono::duration`< \_Rep, \_Period > &\_\_rel) const
- template<typename \_Clock, typename \_Duration >  
bool **wait\_until** (const `chrono::time_point`< \_Clock, \_Duration > &\_\_abs)  
const

### Static Public Member Functions

- template<typename \_Res, typename \_Allocator >  
static `_Ptr`< `_Result_alloc`< \_Res, \_Allocator > >::type **\_S\_allocate\_result**  
(const \_Allocator &\_\_a)

### Protected Types

- typedef `__future_base::_Result`< \_Res > & **\_\_result\_type**
- typedef `shared_ptr`< `_State` > **\_\_state\_type**

### Protected Member Functions

- `__result_type` **\_M\_get\_result** ()
- void **\_M\_swap** (`__basic_future` &\_\_that)

#### 5.693.1 Detailed Description

`template<typename _Res> class std::shared_future< _Res >`

Primary template for `shared_future`.

Definition at line 709 of file `future`.

#### 5.693.2 Constructor & Destructor Documentation

**5.693.2.1** `template<typename _Res > std::shared_future< _Res`  
`>::shared_future ( const shared_future< _Res > & __sf )`  
`[inline]`

Copy constructor.

Definition at line 717 of file `future`.

**5.693.2.2** `template<typename _Res> std::shared_future<_Res>::shared_future ( future<_Res> && __uf ) [inline]`

Construct from a future rvalue.

Definition at line 720 of file `future`.

**5.693.2.3** `template<typename _Res> std::shared_future<_Res>::shared_future ( shared_future<_Res> && __sf ) [inline]`

Construct from a [shared\\_future](#) rvalue.

Definition at line 725 of file `future`.

### 5.693.3 Member Function Documentation

**5.693.3.1** `template<typename _Res> __result_type std::__basic_future<_Res>::__M_get_result ( ) [inline, protected, inherited]`

Wait for the state to be ready and rethrow any stored exception.

Definition at line 538 of file `future`.

Referenced by `std::shared_future<_Res>::get()`, `std::future<void>::get()`, and `std::future<_Res>::get()`.

**5.693.3.2** `template<typename _Res> const _Res& std::shared_future<_Res>::get ( ) [inline]`

Retrieving the value.

Definition at line 743 of file `future`.

References `std::__basic_future<_Res>::__M_get_result()`.

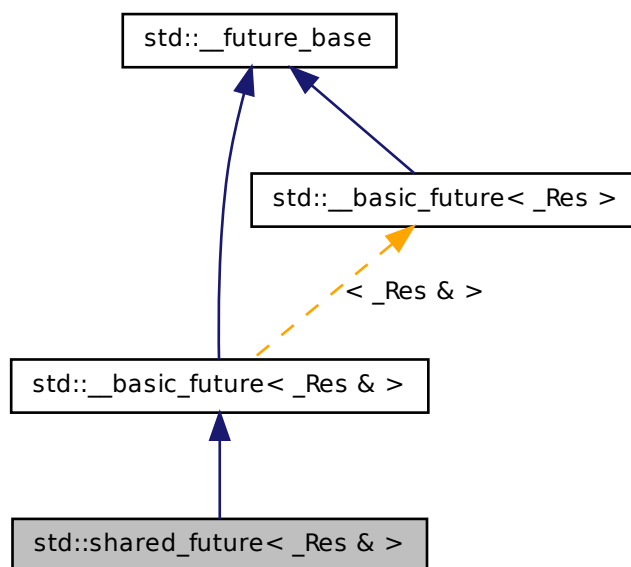
The documentation for this class was generated from the following file:

- [future](#)

**5.694 `std::shared_future<_Res &>` Class Template Reference**

Partial specialization for `shared_future<R&>`

Inheritance diagram for `std::shared_future<_Res &>`:

**Public Member Functions**

- `shared_future` (const `shared_future` &\_\_sf)
- `shared_future` (`shared_future` &&\_\_sf)
- `shared_future` (`future<_Res &>` &&\_\_uf)
- `_Res &` `get` ()
- `shared_future` & `operator=` (const `shared_future` &\_\_sf)
- `shared_future` & `operator=` (`shared_future` &&\_\_sf)
- bool `valid` () const
- void `wait` () const
- bool `wait_for` (const `chrono::duration<_Rep, _Period>` &\_\_rel) const
- bool `wait_until` (const `chrono::time_point<_Clock, _Duration>` &\_\_abs) const



### Static Public Member Functions

- `template<typename _Res, typename _Allocator >  
static \_Ptr< \_Result\_alloc< _Res, _Allocator > >::type \_S\_allocate\_result  
(const _Allocator &__a)`

### Protected Types

- `typedef \_\_future\_base::\_Result< _Res &> & \_\_result\_type`
- `typedef shared\_ptr< \_State > \_\_state\_type`

### Protected Member Functions

- `\_\_result\_type \_M\_get\_result ()`
- `void \_M\_swap (\_\_basic\_future &__that)`

#### 5.694.1 Detailed Description

`template<typename _Res> class std::shared_future< _Res &>`

Partial specialization for `shared_future<R&>`

Definition at line 753 of file `future`.

#### 5.694.2 Constructor & Destructor Documentation

**5.694.2.1** `template<typename _Res > std::shared_future< _Res &  
>::shared_future ( const shared_future< _Res &> & __sf )  
[inline]`

Copy constructor.

Definition at line 761 of file `future`.

**5.694.2.2** `template<typename _Res > std::shared_future< _Res &  
>::shared_future ( future< _Res &> && __uf ) [inline]`

Construct from a future rvalue.

Definition at line 764 of file `future`.

**5.694.2.3** `template<typename _Res > std::shared_future< _Res &  
>::shared_future ( shared_future< _Res & > && __sf )  
[inline]`

Construct from a [shared\\_future](#) rvalue.

Definition at line 769 of file future.

### 5.694.3 Member Function Documentation

**5.694.3.1** `__result_type std::__basic_future< _Res & >::M_get_result ( )  
[inline, protected, inherited]`

Wait for the state to be ready and rethrow any stored exception.

Definition at line 538 of file future.

References `std::rethrow_exception()`.

**5.694.3.2** `template<typename _Res > _Res& std::shared_future< _Res &  
>::get ( ) [inline]`

Retrieving the value.

Definition at line 787 of file future.

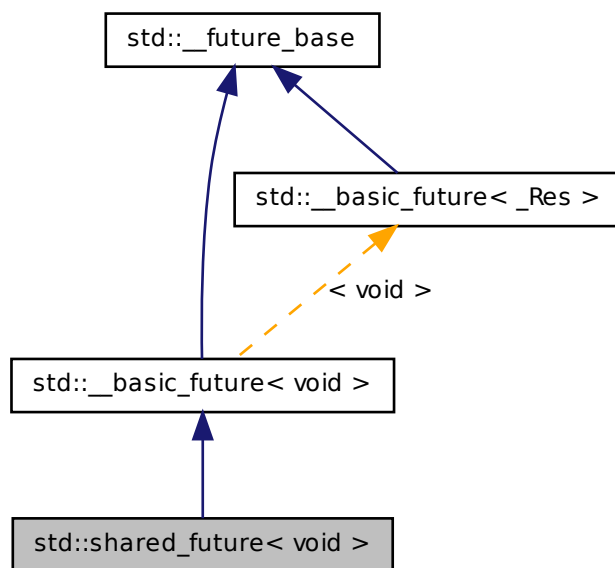
The documentation for this class was generated from the following file:

- [future](#)

## 5.695 `std::shared_future< void >` Class Template Reference

Explicit specialization for [shared\\_future<void>](#)

Inheritance diagram for `std::shared_future< void >`:



### Public Member Functions

- `shared_future` (const `shared_future` &\_\_sf)
- `shared_future` (`shared_future` &&\_\_sf)
- `shared_future` (`future`< void > &&\_\_uf)
- void `get` ()
- `shared_future` & `operator=` (const `shared_future` &\_\_sf)
- `shared_future` & `operator=` (`shared_future` &&\_\_sf)
- bool `valid` () const
- void `wait` () const
- bool `wait_for` (const `chrono::duration`< \_Rep, \_Period > &\_\_rel) const
- bool `wait_until` (const `chrono::time_point`< \_Clock, \_Duration > &\_\_abs) const

**Static Public Member Functions**

- `template<typename _Res, typename _Allocator >`  
`static \_Ptr< \_Result\_alloc< _Res, _Allocator > >::type \_S\_allocate\_result`  
`(const _Allocator &__a)`

**Protected Types**

- `typedef \_\_future\_base::\_Result< void > & \_\_result\_type`
- `typedef shared\_ptr< \_State > \_\_state\_type`

**Protected Member Functions**

- `\_\_result\_type \_M\_get\_result ()`
- `void \_M\_swap (\_\_basic\_future &__that)`

**5.695.1 Detailed Description**

`template<> class std::shared_future< void >`

Explicit specialization for `shared_future<void>`

Definition at line 792 of file future.

**5.695.2 Constructor & Destructor Documentation**

**5.695.2.1** `std::shared_future< void >::shared_future ( const shared_future< void > & __sf ) [inline]`

Copy constructor.

Definition at line 800 of file future.

**5.695.2.2** `std::shared_future< void >::shared_future ( future< void > && __uf ) [inline]`

Construct from a future rvalue.

Definition at line 803 of file future.

### 5.695.2.3 `std::shared_future<void>::shared_future ( shared_future<void> && __sf ) [inline]`

Construct from a [shared\\_future](#) rvalue.

Definition at line 808 of file `future`.

## 5.695.3 Member Function Documentation

### 5.695.3.1 `__result_type std::__basic_future<void>::_M_get_result ( ) [inline, protected, inherited]`

Wait for the state to be ready and rethrow any stored exception.

Definition at line 538 of file `future`.

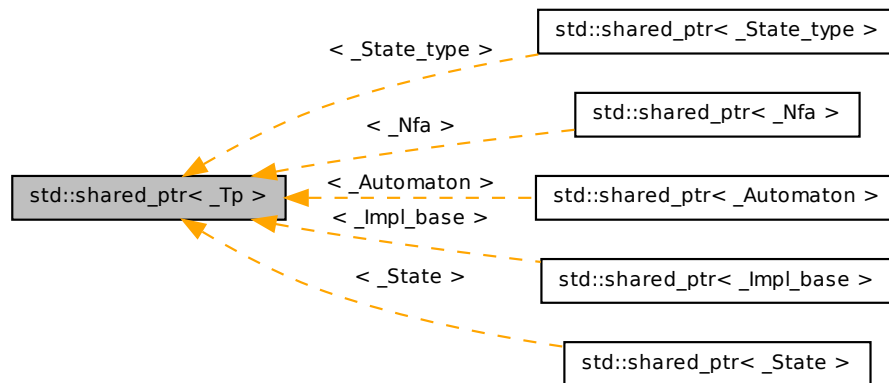
The documentation for this class was generated from the following file:

- [future](#)

## 5.696 `std::shared_ptr<_Tp>` Class Template Reference

A smart pointer with reference-counted copy semantics.

Inheritance diagram for std::shared\_ptr<\_Tp>:



## Public Types

- typedef `_Tp element_type`

## Public Member Functions

- constexpr `shared_ptr()`
- template<typename `_Tp1`>  
`shared_ptr(_Tp1 *__p)`
- template<typename `_Deleter`>  
`shared_ptr(nullptr_t __p, _Deleter __d)`
- template<typename `_Tp1`, typename = typename std::enable\_if<std::is\_convertible<\_Tp1\*, \_Tp\*>::value>::type>  
`shared_ptr(const shared_ptr<_Tp1> &__r)`
- `shared_ptr(shared_ptr &&__r)`
- template<typename `_Tp1`, typename `_Deleter`, typename `_Alloc`>  
`shared_ptr(_Tp1 *__p, _Deleter __d, _Alloc __a)`
- template<typename `_Tp1`, typename = typename std::enable\_if<std::is\_convertible<\_Tp1\*, \_Tp\*>::value>::type>  
`shared_ptr(shared_ptr<_Tp1> &&__r)`
- template<typename `_Tp1`>  
`shared_ptr(const weak_ptr<_Tp1> &__r)`

- `template<typename _Tp1, typename _Deleter>`  
`shared_ptr` (`_Tp1 *__p, _Deleter __d`)
- `template<typename _Deleter, typename _Alloc>`  
`shared_ptr` (`nullptr_t __p, _Deleter __d, _Alloc __a`)
- `template<typename _Tp1, typename _Del>`  
`shared_ptr` (`std::unique_ptr<_Tp1, _Del> &&__r`)
- `constexpr shared_ptr` (`nullptr_t __p`)
- `template<typename _Tp1>`  
`shared_ptr` (`const shared_ptr<_Tp1> &__r, _Tp *__p`)
- `_Tp * get () const`
- `operator bool () const`
- `std::add_lvalue_reference<_Tp>::type operator* () const`
- `_Tp * operator-> () const`
- `template<class _Tp1>`  
`shared_ptr & operator=` (`shared_ptr<_Tp1> &&__r`)
- `template<typename _Tp1, typename _Del>`  
`shared_ptr & operator=` (`std::unique_ptr<_Tp1, _Del> &&__r`)
- `shared_ptr & operator=` (`shared_ptr &&__r`)
- `template<typename _Tp1>`  
`shared_ptr & operator=` (`const shared_ptr<_Tp1> &__r`)
- `template<typename _Tp1>`  
`bool owner_before` (`__weak_ptr<_Tp1, _Lp> const &__rhs`) `const`
- `template<typename _Tp1>`  
`bool owner_before` (`__shared_ptr<_Tp1, _Lp> const &__rhs`) `const`
- `template<typename _Tp1, typename _Deleter>`  
`void reset` (`_Tp1 *__p, _Deleter __d`)
- `template<typename _Tp1, typename _Deleter, typename _Alloc>`  
`void reset` (`_Tp1 *__p, _Deleter __d, _Alloc __a`)
- `template<typename _Tp1>`  
`void reset` (`_Tp1 *__p`)
- `void reset ()`
- `void swap` (`__shared_ptr<_Tp, _Lp> &__other`)
- `bool unique () const`
- `long use_count () const`

## Friends

- `template<typename _Tp1, _Lock_policy _Lp1, typename _Alloc, typename... _Args>`  
`__shared_ptr<_Tp1, _Lp1> __allocate_shared` (`const _Alloc &__a, _Args &&...__args`)
- `template<typename _Tp1, typename _Alloc, typename... _Args>`  
`shared_ptr<_Tp1> allocate_shared` (`const _Alloc &__a, _Args &&...__args`)

### 5.696.1 Detailed Description

**template<typename \_Tp> class std::shared\_ptr<\_Tp>**

A smart pointer with reference-counted copy semantics. The object pointed to is deleted when the last [shared\\_ptr](#) pointing to it is destroyed or reset.

Definition at line 93 of file shared\_ptr.h.

### 5.696.2 Constructor & Destructor Documentation

**5.696.2.1** **template<typename \_Tp> constexpr std::shared\_ptr<\_Tp>  
>::shared\_ptr( ) [inline]**

Construct an empty shared\_ptr.

#### Postcondition

use\_count()==0 && get()==0

Definition at line 100 of file shared\_ptr.h.

**5.696.2.2** **template<typename \_Tp> template<typename \_Tp1>  
std::shared\_ptr<\_Tp>::shared\_ptr( \_Tp1 \* \_\_p ) [inline,  
explicit]**

Construct a shared\_ptr that owns the pointer *\_\_p*.

#### Parameters

*\_\_p* A pointer that is convertible to element\_type\*.

#### Postcondition

use\_count() == 1 && get() == *\_\_p*

#### Exceptions

[std::bad\\_alloc](#), in which case delete *\_\_p* is called.

Definition at line 110 of file shared\_ptr.h.



**5.696.2.3** `template<typename _Tp> template<typename _Tp1, typename  
_Deleter > std::shared_ptr<_Tp>::shared_ptr ( _Tp1 * __p,  
_Deleter __d ) [inline]`

Construct a shared\_ptr that owns the pointer `__p` and the deleter `__d`.

#### Parameters

`__p` A pointer.

`__d` A deleter.

#### Postcondition

`use_count() == 1 && get() == __p`

#### Exceptions

*`std::bad_alloc`*, in which case `__d(__p)` is called.

Requirements: `_Deleter`'s copy constructor and destructor must not throw

`__shared_ptr` will release `__p` by calling `__d(__p)`

Definition at line 127 of file `shared_ptr.h`.

**5.696.2.4** `template<typename _Tp> template<typename _Deleter >  
std::shared_ptr<_Tp>::shared_ptr ( nullptr_t __p, _Deleter __d  
) [inline]`

Construct a shared\_ptr that owns a null pointer and the deleter `__d`.

#### Parameters

`__p` A null pointer constant.

`__d` A deleter.

#### Postcondition

`use_count() == 1 && get() == __p`

#### Exceptions

*`std::bad_alloc`*, in which case `__d(__p)` is called.

Requirements: \_Deleter's copy constructor and destructor must not throw

The last owner will call \_\_d(\_\_p)

Definition at line 144 of file shared\_ptr.h.

**5.696.2.5** `template<typename _Tp> template<typename _Tp1, typename  
_Deleter, typename _Alloc> std::shared_ptr<_Tp>::shared_ptr(  
_Tp1 * __p, _Deleter __d, _Alloc __a) [inline]`

Construct a shared\_ptr that owns the pointer \_\_p and the deleter \_\_d.

#### Parameters

**\_\_p** A pointer.

**\_\_d** A deleter.

**\_\_a** An allocator.

#### Postcondition

use\_count() == 1 && get() == \_\_p

#### Exceptions

*std::bad\_alloc*, in which case \_\_d(\_\_p) is called.

Requirements: \_Deleter's copy constructor and destructor must not throw \_Alloc's copy constructor and destructor must not throw.

\_\_shared\_ptr will release \_\_p by calling \_\_d(\_\_p)

Definition at line 163 of file shared\_ptr.h.

**5.696.2.6** `template<typename _Tp> template<typename _Deleter, typename  
_Alloc> std::shared_ptr<_Tp>::shared_ptr( nullptr_t __p,  
_Deleter __d, _Alloc __a) [inline]`

Construct a shared\_ptr that owns a null pointer and the deleter \_\_d.

#### Parameters

**\_\_p** A null pointer constant.

**\_\_d** A deleter.

**\_\_a** An allocator.

**Postcondition**

use\_count() == 1 && get() == \_\_p

**Exceptions**

[\*std::bad\\_alloc\*](#), in which case `__d(__p)` is called.

Requirements: `_Deleter`'s copy constructor and destructor must not throw `_Alloc`'s copy constructor and destructor must not throw.

The last owner will call `__d(__p)`

Definition at line 182 of file `shared_ptr.h`.

**5.696.2.7** `template<typename _Tp> template<typename _Tp1 >  
std::shared_ptr<_Tp>::shared_ptr ( const shared_ptr<_Tp1 > &  
__r, _Tp * __p ) [inline]`

Constructs a `shared_ptr` instance that stores `__p` and shares ownership with `__r`.

**Parameters**

`__r` A `shared_ptr`.

`__p` A pointer that will remain valid while `*__r` is valid.

**Postcondition**

get() == \_\_p && use\_count() == \_\_r.use\_count()

This can be used to construct a [`shared\_ptr`](#) to a sub-object of an object managed by an existing [`shared\_ptr`](#).

```
shared_ptr< pair<int,int> > pii(new pair<int,int>());
shared_ptr<int> pi(pii, &pii->first);
assert(pii.use_count() == 2);
```

Definition at line 204 of file `shared_ptr.h`.

**5.696.2.8** `template<typename _Tp> template<typename _Tp1, typename  
= typename std::enable_if<std::is_convertible<_Tp1*,  
_Tp*>::value>::type> std::shared_ptr<_Tp>::shared_ptr ( const  
shared_ptr<_Tp1 > & __r ) [inline]`

If `__r` is empty, constructs an empty `shared_ptr`; otherwise construct a `shared_ptr` that shares ownership with `__r`.

#### Parameters

`__r` A `shared_ptr`.

#### Postcondition

`get() == __r.get() && use_count() == __r.use_count()`

Definition at line 216 of file `shared_ptr.h`.

**5.696.2.9** `template<typename _Tp> std::shared_ptr<_Tp>::shared_ptr (`  
`shared_ptr<_Tp> && __r ) [inline]`

Move-constructs a `shared_ptr` instance from `__r`.

#### Parameters

`__r` A `shared_ptr` rvalue.

#### Postcondition

\*this contains the old value of `__r`, `__r` is empty.

Definition at line 224 of file `shared_ptr.h`.

**5.696.2.10** `template<typename _Tp> template<typename _Tp1, typename`  
`= typename std::enable_if<std::is_convertible<_Tp1*,`  
`_Tp*>::value>::type> std::shared_ptr<_Tp>::shared_ptr (`  
`shared_ptr<_Tp1> && __r ) [inline]`

Move-constructs a `shared_ptr` instance from `__r`.

#### Parameters

`__r` A `shared_ptr` rvalue.

#### Postcondition

\*this contains the old value of `__r`, `__r` is empty.

Definition at line 234 of file `shared_ptr.h`.

**5.696.2.11** `template<typename _Tp> template<typename _Tp1 >  
std::shared_ptr<_Tp>::shared_ptr ( const weak_ptr<_Tp1> &  
__r ) [inline, explicit]`

Constructs a shared\_ptr that shares ownership with \_\_r and stores a copy of the pointer stored in \_\_r.

#### Parameters

`__r` A [weak\\_ptr](#).

#### Postcondition

`use_count() == __r.use_count()`

#### Exceptions

[bad\\_weak\\_ptr](#) when \_\_r.expired(), in which case the constructor has no effect.

Definition at line 246 of file shared\_ptr.h.

**5.696.2.12** `template<typename _Tp> constexpr std::shared_ptr<_Tp>  
>::shared_ptr ( nullptr_t __p ) [inline]`

Construct an empty shared\_ptr.

#### Parameters

`__p` A null pointer constant.

#### Postcondition

`use_count() == 0 && get() == nullptr`

Definition at line 264 of file shared\_ptr.h.

### 5.696.3 Friends And Related Function Documentation

**5.696.3.1** `template<typename _Tp> template<typename _Tp1, typename  
_Alloc, typename... _Args> shared_ptr<_Tp1> allocate_shared (  
const _Alloc & __a, _Args &&... __args ) [friend]`

Create an object that is owned by a [shared\\_ptr](#).

### Parameters

- `__a` An allocator.
- `__args` Arguments for the `_Tp` object's constructor.

### Returns

A [shared\\_ptr](#) that owns the newly created object.

### Exceptions

An exception thrown from `_Alloc::allocate` or from the constructor of `_Tp`.

A copy of `__a` will be used to allocate memory for the [shared\\_ptr](#) and the new object.

Definition at line 528 of file `shared_ptr.h`.

The documentation for this class was generated from the following file:

- [shared\\_ptr.h](#)

## 5.697 `std::shuffle_order_engine< _RandomNumberEngine, __k >` Class Template Reference

Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits `__w`.

### Public Types

- typedef `_RandomNumberEngine::result_type` [result\\_type](#)

### Public Member Functions

- [shuffle\\_order\\_engine](#) ()
- [shuffle\\_order\\_engine](#) (const `_RandomNumberEngine &__rne`)
- [shuffle\\_order\\_engine](#) ([result\\_type](#) \_\_s)
- template<typename `_Sseq` , typename = typename `std::enable_if<!std::is_same<_Sseq, shuffle_order_engine>::value && !std::is_same<_Sseq, _RandomNumberEngine>::value> ::type>`  
[shuffle\\_order\\_engine](#) (`_Sseq &__q`)
- [shuffle\\_order\\_engine](#) (`_RandomNumberEngine &&__rne`)
- const `_RandomNumberEngine &` [base](#) () const
- void [discard](#) (unsigned long long `__z`)
- [result\\_type](#) [operator](#)() ()
- void [seed](#) ()
- template<typename `_Sseq` >  
void [seed](#) (`_Sseq &__q`)
- void [seed](#) ([result\\_type](#) \_\_s)

### Static Public Member Functions

- static constexpr `result_type` `max` ()
- static constexpr `result_type` `min` ()

### Static Public Attributes

- static constexpr `size_t` `table_size`

### Friends

- `template<typename _RandomNumberEngine1, size_t __k1, typename _CharT, typename _Traits>`  
`>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &, const std::shuffle_order_engine< _RandomNumberEngine1, __k1 > &)`
- `bool operator== (const shuffle_order_engine &__lhs, const shuffle_order_engine &__rhs)`
- `template<typename _RandomNumberEngine1, size_t __k1, typename _CharT, typename _Traits>`  
`>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &, std::shuffle_order_engine< _RandomNumberEngine1, __k1 > &)`

#### 5.697.1 Detailed Description

`template<typename _RandomNumberEngine, size_t __k> class std::shuffle_order_engine< _RandomNumberEngine, __k >`

Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits `__w`.

Definition at line 1217 of file `random.h`.

#### 5.697.2 Member Typedef Documentation

**5.697.2.1** `template<typename _RandomNumberEngine, size_t __k> typedef _RandomNumberEngine::result_type std::shuffle_order_engine< _RandomNumberEngine, __k >::result_type`

The type of the generated random value.

Definition at line 1224 of file `random.h`.

### 5.697.3 Constructor & Destructor Documentation

**5.697.3.1** `template<typename _RandomNumberEngine, size_t __k>  
std::shuffle_order_engine< _RandomNumberEngine, __k  
>::shuffle_order_engine ( ) [inline]`

Constructs a default shuffle\_order\_engine engine.

The underlying engine is default constructed as well.

Definition at line 1233 of file random.h.

**5.697.3.2** `template<typename _RandomNumberEngine, size_t __k>  
std::shuffle_order_engine< _RandomNumberEngine, __k  
>::shuffle_order_engine ( const _RandomNumberEngine & __rne )  
[inline, explicit]`

Copy constructs a shuffle\_order\_engine engine.

Copies an existing base class random number generator.

#### Parameters

*rne* An existing (base class) engine object.

Definition at line 1244 of file random.h.

**5.697.3.3** `template<typename _RandomNumberEngine, size_t __k>  
std::shuffle_order_engine< _RandomNumberEngine, __k  
>::shuffle_order_engine ( _RandomNumberEngine && __rne )  
[inline, explicit]`

Move constructs a shuffle\_order\_engine engine.

Copies an existing base class random number generator.

#### Parameters

*rne* An existing (base class) engine object.

Definition at line 1255 of file random.h.



**5.697.3.4** template<typename \_RandomNumberEngine, size\_t \_\_k>  
std::shuffle\_order\_engine< \_RandomNumberEngine, \_\_k  
>::shuffle\_order\_engine ( result\_type \_\_s ) [inline,  
explicit]

Seed constructs a shuffle\_order\_engine engine.

Constructs the underlying generator engine seeded with \_\_s.

#### Parameters

\_\_s A seed value for the base class engine.

Definition at line 1266 of file random.h.

**5.697.3.5** template<typename \_RandomNumberEngine, size\_t  
\_\_k> template<typename \_Sseq, typename = typename  
std::enable\_if<!std::is\_same<\_Sseq, shuffle\_order\_engine>::value  
&& !std::is\_same<\_Sseq, \_RandomNumberEngine>::value>  
::type> std::shuffle\_order\_engine< \_RandomNumberEngine, \_\_k  
>::shuffle\_order\_engine ( \_Sseq & \_\_q ) [inline, explicit]

Generator construct a shuffle\_order\_engine engine.

#### Parameters

\_\_q A seed sequence.

Definition at line 1280 of file random.h.

### 5.697.4 Member Function Documentation

**5.697.4.1** template<typename \_RandomNumberEngine, size\_t \_\_k>  
const \_RandomNumberEngine& std::shuffle\_order\_engine<  
\_RandomNumberEngine, \_\_k >::base ( ) const [inline]

Gets a const reference to the underlying generator engine object.

Definition at line 1323 of file random.h.

**5.697.4.2** `template<typename _RandomNumberEngine, size_t __k> void  
std::shuffle_order_engine<_RandomNumberEngine, __k>::discard  
( unsigned long long __z ) [inline]`

Discard a sequence of random numbers.

Definition at line 1344 of file random.h.

**5.697.4.3** `template<typename _RandomNumberEngine, size_t __k>  
static constexpr result_type std::shuffle_order_engine<  
_RandomNumberEngine, __k>::max( ) [inline, static]`

Gets the maximum value in the generated random number range.

Definition at line 1337 of file random.h.

**5.697.4.4** `template<typename _RandomNumberEngine, size_t __k>  
static constexpr result_type std::shuffle_order_engine<  
_RandomNumberEngine, __k>::min( ) [inline, static]`

Gets the minimum value in the generated random number range.

Definition at line 1330 of file random.h.

**5.697.4.5** `template<typename _RandomNumberEngine, size_t __k>  
shuffle_order_engine<_RandomNumberEngine, __k>::result_type  
std::shuffle_order_engine<_RandomNumberEngine, __k  
>::operator()( )`

Gets the next value in the generated random number sequence.

Definition at line 775 of file random.tcc.

**5.697.4.6** `template<typename _RandomNumberEngine, size_t __k>  
template<typename _Sseq> void std::shuffle_order_engine<  
_RandomNumberEngine, __k>::seed( _Sseq & __q ) [inline]`

Reseeds the `shuffle_order_engine` object with the given seed sequence.

#### Parameters

`__q` A seed generator function.

Definition at line 1313 of file random.h.

**5.697.4.7** `template<typename _RandomNumberEngine, size_t __k> void  
std::shuffle_order_engine< _RandomNumberEngine, __k >::seed (  
result_type __s ) [inline]`

Reseeds the shuffle\_order\_engine object with the default seed for the underlying base class generator engine.

Definition at line 1300 of file random.h.

**5.697.4.8** `template<typename _RandomNumberEngine, size_t __k> void  
std::shuffle_order_engine< _RandomNumberEngine, __k >::seed (  
) [inline]`

Reseeds the shuffle\_order\_engine object with the default seed for the underlying base class generator engine.

Definition at line 1289 of file random.h.

## 5.697.5 Friends And Related Function Documentation

**5.697.5.1** `template<typename _RandomNumberEngine, size_t __k>  
template<typename _RandomNumberEngine1 , size_t __k1,  
typename _CharT , typename _Traits > std::basic_ostream< _CharT,  
_Traits>& operator<< ( std::basic_ostream< _CharT, _Traits > & ,  
const std::shuffle_order_engine< _RandomNumberEngine1, __k1 >  
& ) [friend]`

Inserts the current state of a shuffle\_order\_engine random number generator engine \_\_x into the output stream \_\_os.

### Parameters

`__os` An output stream.

`__x` A shuffle\_order\_engine random number generator engine.

### Returns

The output stream with the state of \_\_x inserted or in an error state.

**5.697.5.2** `template<typename _RandomNumberEngine, size_t __k> bool  
operator==( const shuffle_order_engine< _RandomNumberEngine,  
__k > & __lhs, const shuffle_order_engine<  
_RandomNumberEngine, __k > & __rhs ) [friend]`

Compares two shuffle\_order\_engine random number generator objects of the same type for equality.

#### Parameters

`__lhs` A shuffle\_order\_engine random number generator object.

`__rhs` Another shuffle\_order\_engine random number generator object.

#### Returns

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 1368 of file random.h.

**5.697.5.3** `template<typename _RandomNumberEngine, size_t __k>  
template<typename _RandomNumberEngine1, size_t __k1,  
typename _CharT, typename _Traits > std::basic_istream<_CharT,  
_Traits>& operator>> ( std::basic_istream<_CharT, _Traits > &,  
std::shuffle_order_engine< _RandomNumberEngine1, __k1 > & )  
[friend]`

Extracts the current state of a % subtract\_with\_carry\_engine random number generator engine `__x` from the input stream `__is`.

#### Parameters

`__is` An input stream.

`__x` A shuffle\_order\_engine random number generator engine.

#### Returns

The input stream with the state of `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.698 `std::slice` Class Reference

Class defining one-dimensional subset of an array.

### Public Member Functions

- [slice](#) ()
- [slice](#) (size\_t, size\_t, size\_t)
- size\_t [size](#) () const
- size\_t [start](#) () const
- size\_t [stride](#) () const

### 5.698.1 Detailed Description

Class defining one-dimensional subset of an array. The slice class represents a one-dimensional subset of an array, specified by three parameters: start offset, size, and stride. The start offset is the index of the first element of the array that is part of the subset. The size is the total number of elements in the subset. Stride is the distance between each successive array element to include in the subset.

For example, with an array of size 10, and a slice with offset 1, size 3 and stride 2, the subset consists of array elements 1, 3, and 5.

Definition at line 60 of file `slice_array.h`.

The documentation for this class was generated from the following file:

- [slice\\_array.h](#)

## 5.699 `std::slice_array<_Tp>` Class Template Reference

Reference to one-dimensional subset of an array.

### Public Types

- typedef `_Tp` **value\_type**

### Public Member Functions

- [slice\\_array](#) (const [slice\\_array](#) &)
- template<class `_Dom` >  
void **operator%=** (const `_Expr`< `_Dom`, `_Tp` > &) const
- void [operator%=](#) (const [valarray](#)< `_Tp` > &) const

- void **operator&=** (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void **operator&=** (const \_Expr< \_Dom, \_Tp > &) const
- template<class \_Dom >  
void **operator\*=** (const \_Expr< \_Dom, \_Tp > &) const
- void **operator\*=** (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void **operator+=** (const \_Expr< \_Dom, \_Tp > &) const
- void **operator+=** (const [valarray](#)< \_Tp > &) const
- void **operator-=** (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void **operator-=** (const \_Expr< \_Dom, \_Tp > &) const
- template<class \_Dom >  
void **operator/=** (const \_Expr< \_Dom, \_Tp > &) const
- void **operator/=** (const [valarray](#)< \_Tp > &) const
- void **operator<<=** (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void **operator<<=** (const \_Expr< \_Dom, \_Tp > &) const
- void **operator=** (const \_Tp &) const
- [slice\\_array](#) & **operator=** (const [slice\\_array](#) &)
- template<class \_Dom >  
void **operator=** (const \_Expr< \_Dom, \_Tp > &) const
- void **operator=** (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void **operator>>=** (const \_Expr< \_Dom, \_Tp > &) const
- void **operator>>=** (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void **operator^=** (const \_Expr< \_Dom, \_Tp > &) const
- void **operator^=** (const [valarray](#)< \_Tp > &) const
- void **operator|=** (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void **operator|=** (const \_Expr< \_Dom, \_Tp > &) const

## Friends

- class [valarray](#)< \_Tp >

### 5.699.1 Detailed Description

**template<typename \_Tp> class std::slice\_array< \_Tp >**

Reference to one-dimensional subset of an array. A [slice\\_array](#) is a reference to the actual elements of an array specified by a slice. The way to get a [slice\\_array](#) is to call

operator[ ](slice) on a valarray. The returned [slice\\_array](#) then permits carrying operations out on the referenced subset of elements in the original valarray. For example, operator+=(valarray) will add values to the subset of elements in the underlying valarray this [slice\\_array](#) refers to.

### Parameters

*Tp* Element type.

Definition at line 124 of file slice\_array.h.

## 5.699.2 Member Function Documentation

**5.699.2.1** `template<typename _Tp> void std::slice_array< _Tp >::operator%=( const valarray< _Tp > & ) const`

Modulo slice elements by corresponding elements of *v*.

**5.699.2.2** `template<typename _Tp> void std::slice_array< _Tp >::operator&=( const valarray< _Tp > & ) const`

Logical and slice elements with corresponding elements of *v*.

**5.699.2.3** `template<typename _Tp> void std::slice_array< _Tp >::operator*=( const valarray< _Tp > & ) const`

Multiply slice elements by corresponding elements of *v*.

**5.699.2.4** `template<typename _Tp> void std::slice_array< _Tp >::operator+=( const valarray< _Tp > & ) const`

Add corresponding elements of *v* to slice elements.

**5.699.2.5** `template<typename _Tp> void std::slice_array< _Tp >::operator-=( const valarray< _Tp > & ) const`

Subtract corresponding elements of  $v$  from slice elements.

**5.699.2.6** `template<typename _Tp> void std::slice_array<_Tp>::operator/=`  
`( const valarray<_Tp> & ) const`

Divide slice elements by corresponding elements of  $v$ .

**5.699.2.7** `template<typename _Tp> void std::slice_array<_Tp>`  
`>::operator<<= ( const valarray<_Tp> & ) const`

Left shift slice elements by corresponding elements of  $v$ .

**5.699.2.8** `template<typename _Tp> void std::slice_array<_Tp>`  
`>::operator>>= ( const valarray<_Tp> & ) const`

Right shift slice elements by corresponding elements of  $v$ .

**5.699.2.9** `template<typename _Tp> void std::slice_array<_Tp>::operator^=`  
`( const valarray<_Tp> & ) const`

Logical xor slice elements with corresponding elements of  $v$ .

**5.699.2.10** `template<typename _Tp> void std::slice_array<_Tp>`  
`>::operator|= ( const valarray<_Tp> & ) const`

Logical or slice elements with corresponding elements of  $v$ .

The documentation for this class was generated from the following file:

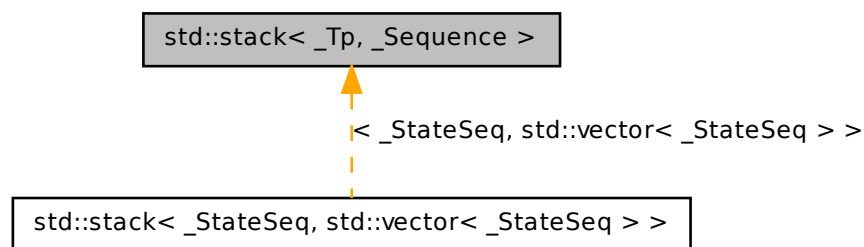
- [slice\\_array.h](#)

## 5.700 `std::stack<_Tp, _Sequence>` Class Template Reference

A standard container giving FILO behavior.



Inheritance diagram for std::stack<\_Tp, \_Sequence>:



### Public Types

- typedef `_Sequence::const_reference` **const\_reference**
- typedef `_Sequence` **container\_type**
- typedef `_Sequence::reference` **reference**
- typedef `_Sequence::size_type` **size\_type**
- typedef `_Sequence::value_type` **value\_type**

### Public Member Functions

- `stack` (`const _Sequence &__c`)
- `stack` (`_Sequence &&__c=_Sequence()`)
- `template<typename... _Args>`  
void **emplace** (`_Args &&...__args`)
- bool `empty` () const
- void `pop` ()
- void `push` (`value_type &&__x`)
- void `push` (`const value_type &__x`)
- `size_type` `size` () const
- void `swap` (`stack &__s`)
- `const_reference` `top` () const
- `reference` `top` ()

**Protected Attributes**

- `_Sequence c`

**Friends**

- `template<typename _Tp1, typename _Seq1 >`  
`bool operator< (const stack< _Tp1, _Seq1 > &, const stack< _Tp1, _Seq1 >`  
`&)`
- `template<typename _Tp1, typename _Seq1 >`  
`bool operator== (const stack< _Tp1, _Seq1 > &, const stack< _Tp1, _Seq1 >`  
`&)`

**5.700.1 Detailed Description**

`template<typename _Tp, typename _Sequence = deque<_Tp>> class std::stack<_Tp, _Sequence>`

A standard container giving FILO behavior. Meets many of the requirements of a [container](#), but does not define anything to do with iterators. Very few of the other standard container interfaces are defined.

This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces strict first-in-last-out stack behavior.

The second template parameter defines the type of the underlying sequence/container. It defaults to [std::deque](#), but it can be any type that supports `back`, `push_back`, and `pop_front`, such as [std::list](#), [std::vector](#), or an appropriate user-defined type.

Members not found in *normal* containers are `container_type`, which is a typedef for the second Sequence parameter, and `push`, `pop`, and `top`, which are standard stack/FILO operations.

Definition at line 95 of file `stl_stack.h`.

**5.700.2 Constructor & Destructor Documentation**

**5.700.2.1** `template<typename _Tp, typename _Sequence = deque<_Tp>>`  
`std::stack<_Tp, _Sequence>::stack ( const _Sequence & __c )`  
`[inline, explicit]`

Default constructor creates no elements.

Definition at line 133 of file `stl_stack.h`.

### 5.700.3 Member Function Documentation

**5.700.3.1** `template<typename _Tp, typename _Sequence = deque<_Tp>>  
bool std::stack<_Tp, _Sequence>::empty ( ) const [inline]`

Returns true if the stack is empty.

Definition at line 145 of file stl\_stack.h.

**5.700.3.2** `template<typename _Tp, typename _Sequence = deque<_Tp>>  
void std::stack<_Tp, _Sequence>::pop ( ) [inline]`

Removes first element.

This is a typical stack operation. It shrinks the stack by one. The time complexity of the operation depends on the underlying sequence.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop()` is called.

Definition at line 211 of file stl\_stack.h.

**5.700.3.3** `template<typename _Tp, typename _Sequence = deque<_Tp>>  
void std::stack<_Tp, _Sequence>::push ( const value_type & __x )  
[inline]`

Add data to the top of the stack.

#### Parameters

*x* Data to be added.

This is a typical stack operation. The function creates an element at the top of the stack and assigns the given data to it. The time complexity of the operation depends on the underlying sequence.

Definition at line 185 of file stl\_stack.h.

**5.700.3.4** `template<typename _Tp, typename _Sequence = deque<_Tp>>  
size_type std::stack<_Tp, _Sequence>::size ( ) const [inline]`

Returns the number of elements in the stack.

Definition at line 150 of file stl\_stack.h.

## 5.701 `std::student_t_distribution<_RealType>` Class Template Reference 3281

**5.700.3.5** `template<typename _Tp, typename _Sequence = deque<_Tp>>  
reference std::stack<_Tp, _Sequence>::top ( ) [inline]`

Returns a read/write reference to the data at the first element of the stack.

Definition at line 158 of file `stl_stack.h`.

**5.700.3.6** `template<typename _Tp, typename _Sequence = deque<_Tp>>  
const_reference std::stack<_Tp, _Sequence>::top ( ) const  
[inline]`

Returns a read-only (constant) reference to the data at the first element of the stack.

Definition at line 169 of file `stl_stack.h`.

The documentation for this class was generated from the following file:

- [stl\\_stack.h](#)

## 5.701 `std::student_t_distribution<_RealType>` Class Template Reference

A [student\\_t\\_distribution](#) random number distribution.

### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_RealType` [result\\_type](#)

### Public Member Functions

- `student_t_distribution` (`_RealType __n=_RealType(1)`)
- `student_t_distribution` (const [param\\_type](#) &\_\_p)
- [result\\_type](#) `max` () const
- [result\\_type](#) `min` () const
- `_RealType` `n` () const
- `template<typename _UniformRandomNumberGenerator>`  
[result\\_type](#) `operator()` (`_UniformRandomNumberGenerator &__urng`)
- `template<typename _UniformRandomNumberGenerator>`  
[result\\_type](#) `operator()` (`_UniformRandomNumberGenerator &__urng, const`  
`param\_type &__p`)

## 5.701 `std::student_t_distribution<_RealType>` Class Template Reference 3282

- void `param` (const `param_type` &\_\_param)
- `param_type param` () const
- void `reset` ()

### Friends

- template<typename \_RealType1, typename \_CharT, typename \_Traits >  
`std::basic_ostream<_CharT, _Traits>` & `operator<<` (`std::basic_ostream<_CharT, _Traits>` &, const `std::student_t_distribution<_RealType1>` &)
- template<typename \_RealType1 >  
bool `operator==` (const `std::student_t_distribution<_RealType1>` &\_\_d1, const `std::student_t_distribution<_RealType1>` &\_\_d2)
- template<typename \_RealType1, typename \_CharT, typename \_Traits >  
`std::basic_istream<_CharT, _Traits>` & `operator>>` (`std::basic_istream<_CharT, _Traits>` &, `std::student_t_distribution<_RealType1>` &)

### 5.701.1 Detailed Description

**template<typename \_RealType = double> class `std::student_t_distribution<_RealType>`**

A `student_t_distribution` random number distribution. The formula for the normal probability mass function is:

$$p(x|n) = \frac{1}{\sqrt{(n\pi)}} \frac{\Gamma((n+1)/2)}{\Gamma(n/2)} \left(1 + \frac{x^2}{n}\right)^{-(n+1)/2}$$

Definition at line 3038 of file random.h.

### 5.701.2 Member Typedef Documentation

**5.701.2.1 template<typename \_RealType = double> typedef `_RealType std::student_t_distribution<_RealType>::result_type`**

The type of the range of the distribution.

Definition at line 3045 of file random.h.

### **5.701.3 Member Function Documentation**

**5.701.3.1** `template<typename _RealType = double> result_type  
std::student_t_distribution< _RealType >::max ( ) const  
[inline]`

Returns the least upper bound value of the distribution.

Definition at line 3121 of file random.h.

**5.701.3.2** `template<typename _RealType = double> result_type  
std::student_t_distribution< _RealType >::min ( ) const  
[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 3114 of file random.h.

**5.701.3.3** `template<typename _RealType = double> template<typename  
_UniformRandomNumberGenerator > result_type  
std::student_t_distribution< _RealType >::operator() (   
_UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 3129 of file random.h.

References std::sqrt().

**5.701.3.4** `template<typename _RealType = double> param_type  
std::student_t_distribution< _RealType >::param ( ) const  
[inline]`

Returns the parameter set of the distribution.

Definition at line 3099 of file random.h.

## 5.701 `std::student_t_distribution<_RealType>` Class Template Reference 3284

**5.701.3.5** `template<typename _RealType = double> void  
std::student_t_distribution<_RealType>::param ( const  
param_type & __param ) [inline]`

Sets the parameter set of the distribution.

### Parameters

`__param` The new parameter set of the distribution.

Definition at line 3107 of file random.h.

**5.701.3.6** `template<typename _RealType = double> void  
std::student_t_distribution<_RealType>::reset ( ) [inline]`

Resets the distribution state.

Definition at line 3082 of file random.h.

References `std::gamma_distribution<_RealType>::reset()`, and `std::normal_distribution<_RealType>::reset()`.

## 5.701.4 Friends And Related Function Documentation

**5.701.4.1** `template<typename _RealType = double> template<typename  
_RealType1 , typename _CharT , typename _Traits >  
std::basic_ostream<_CharT, _Traits>& operator<<  
( std::basic_ostream<_CharT, _Traits> & , const  
std::student_t_distribution<_RealType1> & ) [friend]`

Inserts a `student_t_distribution` random number distribution `__x` into the output stream `__os`.

### Parameters

`__os` An output stream.

`__x` A `student_t_distribution` random number distribution.

### Returns

The output stream with the state of `__x` inserted or in an error state.

## 5.702 `std::student_t_distribution<_RealType>::param_type` Struct Reference 3285

**5.701.4.2** `template<typename _RealType = double> template<typename  
_RealType1 > bool operator==( const std::student_t_distribution<  
_RealType1 > & __d1, const std::student_t_distribution<  
_RealType1 > & __d2 ) [friend]`

Return true if two Student t distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 3151 of file random.h.

**5.701.4.3** `template<typename _RealType = double> template<typename  
_RealType1 , typename _CharT , typename _Traits >  
std::basic_istream<_CharT, _Traits>& operator>>  
( std::basic_istream<_CharT, _Traits > & ,  
std::student_t_distribution<_RealType1 > & ) [friend]`

Extracts a `student_t_distribution` random number distribution `__x` from the input stream `__is`.

### Parameters

- `__is` An input stream.
- `__x` A `student_t_distribution` random number generator engine.

### Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

## 5.702 `std::student_t_distribution<_RealType>::param_type` Struct Reference

### Public Types

- typedef [student\\_t\\_distribution<\\_RealType>](#) `distribution_type`

### Public Member Functions

- `param_type` (`_RealType __n=_RealType(1)`)
- `_RealType n` () const



## 5.702 `std::student_t_distribution<_RealType>::param_type` Struct Reference 3286

---

### Friends

- `bool operator==(const param\_type &__p1, const param\_type &__p2)`

### 5.702.1 Detailed Description

**template<typename `_RealType` = double> struct `std::student_t_distribution<_RealType>::param_type`**

Parameter type.

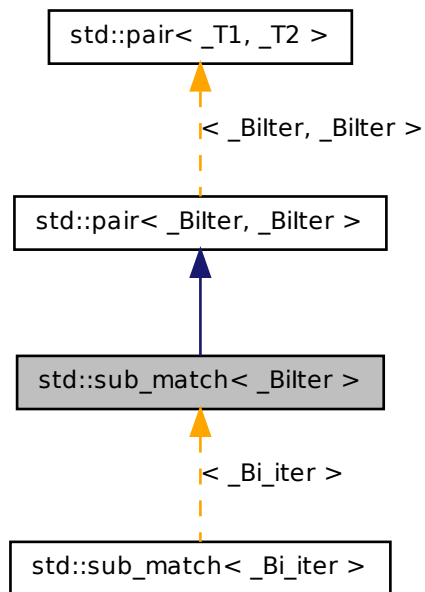
Definition at line 3047 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 5.703 std::sub\_match<\_Biliter> Class Template Reference

Inheritance diagram for std::sub\_match<\_Biliter>:



### Public Types

- typedef `iterator_traits<_Biliter>::difference_type` **difference\_type**
- typedef `_Biliter` **first\_type**
- typedef `_Biliter` **iterator**
- typedef `_Biliter` **second\_type**
- typedef `std::basic_string<value_type>` **string\_type**
- typedef `iterator_traits<_Biliter>::value_type` **value\_type**

### Public Member Functions

- int **compare** (const `sub_match` &\_\_s) const
- int **compare** (const `string_type` &\_\_s) const

- `int compare (const value_type *__s) const`
- `difference_type length () const`
- `operator string_type () const`
- `string_type str () const`
- `void swap (pair &__p)`

#### Public Attributes

- `_BiIter first`
- `bool matched`
- `_BiIter second`

#### 5.703.1 Detailed Description

`template<typename _BiIter> class std::sub_match<_BiIter>`

A sequence of characters matched by a particular marked sub-expression.

An object of this class is essentially a pair of iterators marking a matched subexpression within a regular expression pattern match. Such objects can be converted to and compared with `std::basic_string` objects of a similar base character type as the pattern matched by the regular expression.

The iterators that make up the pair are the usual half-open interval referencing the actual original pattern matched.

Definition at line 756 of file `regex.h`.

#### 5.703.2 Member Typedef Documentation

**5.703.2.1** `typedef _BiIter std::pair<_BiIter, _BiIter>::second_type`  
`[inherited]`

`first_type` is the first bound type

Definition at line 90 of file `stl_pair.h`.

#### 5.703.3 Member Function Documentation

**5.703.3.1** `template<typename _BiIter> int std::sub_match<_BiIter>::compare ( const sub_match<_BiIter> & __s ) const`  
`[inline]`

Compares this and another matched sequence.

#### Parameters

*s* Another matched sequence to compare to this one.

#### Return values

`<0` this matched sequence will collate before *s*.

`=0` this matched sequence is equivalent to *s*.

`>0` this matched sequence will collate after *s*.

Definition at line 815 of file `regex.h`.

Referenced by `std::operator!=()`, `std::operator==()`, `std::operator>()`, and `std::operator>=()`.

**5.703.3.2** `template<typename _Bilter> int std::sub_match<_Bilter>::compare ( const string_type & __s ) const [inline]`

Compares this `sub_match` to a string.

#### Parameters

*s* A string to compare to this `sub_match`.

#### Return values

`<0` this matched sequence will collate before *s*.

`=0` this matched sequence is equivalent to *s*.

`<0` this matched sequence will collate after *s*.

Definition at line 828 of file `regex.h`.

**5.703.3.3** `template<typename _Bilter> int std::sub_match<_Bilter>::compare ( const value_type * __s ) const [inline]`

Compares this `sub_match` to a C-style string.

#### Parameters

*s* A C-style string to compare to this `sub_match`.

**Return values**

- <0 this matched sequence will collate before *s*.
- =0 this matched sequence is equivalent to *s*.
- >0 this matched sequence will collate after *s*.

Definition at line 841 of file regex.h.

**5.703.3.4 template<typename \_Bilter> difference\_type std::sub\_match<\_Bilter>::length ( ) const [inline]**

Gets the length of the matching sequence.

Definition at line 772 of file regex.h.

**5.703.3.5 template<typename \_Bilter> std::sub\_match<\_Bilter>::operator string\_type ( ) const [inline]**

Gets the matching sequence as a string.

**Returns**

the matching sequence as a string.

This is the implicit conversion operator. It is identical to the [str\(\)](#) member function except that it will want to pop up in unexpected places and cause a great deal of confusion and cursing from the unwary.

Definition at line 785 of file regex.h.

**5.703.3.6 template<typename \_Bilter> string\_type std::sub\_match<\_Bilter>::str ( ) const [inline]**

Gets the matching sequence as a string.

**Returns**

the matching sequence as a string.

Definition at line 798 of file regex.h.

Referenced by `std::sub_match<_Bi_iter>::compare()`, `std::operator!=()`, `std::operator<()`, `std::operator<=()`, `std::operator==()`, `std::operator>()`, and `std::operator>=()`.

### 5.703.4 Member Data Documentation

#### 5.703.4.1 `_BiIter std::pair<_BiIter, _BiIter>::first` [inherited]

`second_type` is the second bound type

Definition at line 92 of file `stl_pair.h`.

Referenced by `std::sub_match<_Bi_iter>::length()`, `std::sub_match<_Bi_iter>::operator string_type()`, and `std::sub_match<_Bi_iter>::str()`.

#### 5.703.4.2 `_BiIter std::pair<_BiIter, _BiIter>::second` [inherited]

`first` is a copy of the first object

Definition at line 93 of file `stl_pair.h`.

Referenced by `std::sub_match<_Bi_iter>::length()`, `std::sub_match<_Bi_iter>::operator string_type()`, and `std::sub_match<_Bi_iter>::str()`.

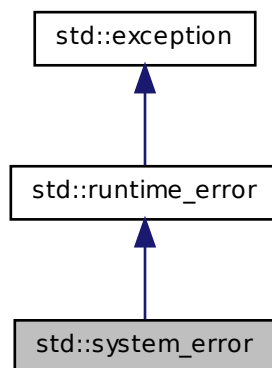
The documentation for this class was generated from the following file:

- [regex.h](#)

## 5.704 `std::system_error` Class Reference

Thrown to indicate error code of underlying system.

Inheritance diagram for std::system\_error:



### Public Member Functions

- **system\_error** ([error\\_code](#) \_\_ec=[error\\_code](#)())
- **system\_error** ([error\\_code](#) \_\_ec, const [string](#) &\_\_what)
- **system\_error** (int \_\_v, const [error\\_category](#) &\_\_ecat, const [string](#) &\_\_what)
- **system\_error** (int \_\_v, const [error\\_category](#) &\_\_ecat)
- const [error\\_code](#) & **code** () const throw ()
- virtual const char \* **what** () const throw ()

#### 5.704.1 Detailed Description

Thrown to indicate error code of underlying system.

Definition at line 309 of file system\_error.

#### 5.704.2 Member Function Documentation

##### 5.704.2.1 virtual const char\* std::runtime\_error::what ( ) const throw () [virtual, inherited]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [system\\_error](#)

## 5.705 `std::thread` Class Reference

`thread`

### Classes

- class [id](#)  
*[thread::id](#)*

### Public Types

- typedef [shared\\_ptr](#)< [\\_Impl\\_base](#) > **\_\_shared\_base\_type**
- typedef `__gthread_t` **native\_handle\_type**

### Public Member Functions

- **thread** ([thread](#) &)
- **thread** ([thread](#) &&\_\_t)
- template<typename [\\_Callable](#) , typename... [\\_Args](#)>  
  **thread** ([\\_Callable](#) &&\_\_f, [\\_Args](#) &&...\_\_args)
- **thread** (const [thread](#) &)
- void **detach** ()
- [thread::id](#) **get\_id** () const
- void **join** ()
- bool **joinable** () const
- `native_handle_type` **native\_handle** ()
- [thread](#) & **operator=** (const [thread](#) &)
- [thread](#) & **operator=** ([thread](#) &&\_\_t)
- void **swap** ([thread](#) &&\_\_t)

### Static Public Member Functions

- static unsigned int **hardware\_concurrency** ()



### 5.705.1 Detailed Description

`thread`

Definition at line 62 of file `thread`.

### 5.705.2 Member Function Documentation

#### 5.705.2.1 `native_handle_type` `std::thread::native_handle ( )` [`inline`]

#### Precondition

`thread` is joinable

Definition at line 177 of file `thread`.

The documentation for this class was generated from the following file:

- [thread](#)

## 5.706 `std::thread::id` Class Reference

[thread::id](#)

### Public Member Functions

- `id` (`native_handle_type __id`)

### Friends

- class `hash< thread::id >`
- bool `operator<` ([thread::id \\_\\_x](#), [thread::id \\_\\_y](#))
- template<class `_CharT`, class `_Traits` >  
[basic\\_ostream< \\_CharT, \\_Traits >](#) & `operator<<` ([basic\\_ostream< \\_CharT, \\_Traits >](#) & `__out`, [thread::id \\_\\_id](#))
- bool `operator==` ([thread::id \\_\\_x](#), [thread::id \\_\\_y](#))
- class `thread`

### 5.706.1 Detailed Description

[thread::id](#)

Definition at line 70 of file thread.

The documentation for this class was generated from the following file:

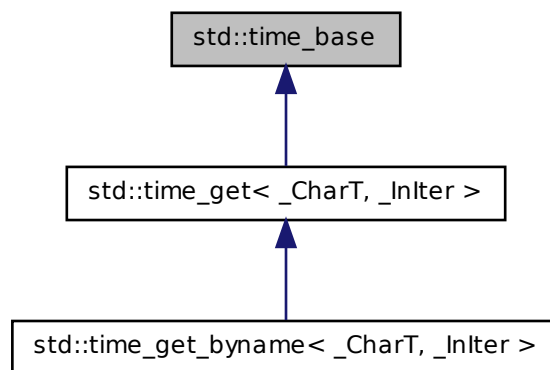
- [thread](#)

## 5.707 std::time\_base Class Reference

Time format ordering data.

This class provides an enum representing different orderings of time: day, month, and year.

Inheritance diagram for std::time\_base:



### Public Types

- enum **dateorder** {  
    **no\_order**, **dmy**, **mdy**, **ymd**,  
    **ydm** }

#### 5.707.1 Detailed Description

Time format ordering data.

This class provides an enum representing different orderings of time: day, month, and year.

Definition at line 52 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following file:

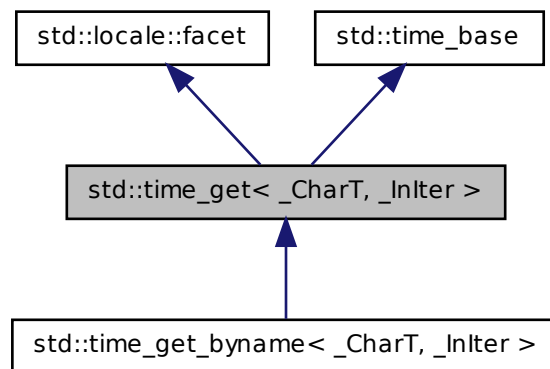
- [locale\\_facets\\_nonio.h](#)

## 5.708 `std::time_get<_CharT, _InIter>` Class Template Reference

Primary class template [time\\_get](#).

This facet encapsulates the code to parse and return a date or time from a string. It is used by the istream numeric extraction operators.

Inheritance diagram for `std::time_get<_CharT, _InIter>`:



### Public Types

- typedef [basic\\_string](#)<\_CharT> **\_\_string\_type**
- enum **dateorder** {  
    **no\_order**, **dmy**, **mdy**, **ydm**,  
    **ydm** }
- typedef \_CharT [char\\_type](#)

- typedef [\\_InIter](#) [iter\\_type](#)

### Public Member Functions

- [time\\_get](#) (size\_t \_\_refs=0)
- dateorder [date\\_order](#) () const
- [iter\\_type](#) [get\\_date](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const
- [iter\\_type](#) [get\\_monthname](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const
- [iter\\_type](#) [get\\_time](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const
- [iter\\_type](#) [get\\_weekday](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const
- [iter\\_type](#) [get\\_year](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const

### Static Public Attributes

- static [locale::id](#) [id](#)

### Protected Member Functions

- virtual [~time\\_get](#) ()
- [iter\\_type](#) [\\_M\\_extract\\_name](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, int &\_\_member, const \_CharT \*\*\_\_names, size\_t \_\_indexlen, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err) const
- [iter\\_type](#) [\\_M\\_extract\\_num](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, int &\_\_member, int \_\_min, int \_\_max, size\_t \_\_len, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err) const
- [iter\\_type](#) [\\_M\\_extract\\_via\\_format](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm, const \_CharT \*\_\_format) const
- [iter\\_type](#) [\\_M\\_extract\\_wday\\_or\\_month](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, int &\_\_member, const \_CharT \*\*\_\_names, size\_t \_\_indexlen, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err) const
- virtual dateorder [do\\_date\\_order](#) () const
- virtual [iter\\_type](#) [do\\_get\\_date](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const
- virtual [iter\\_type](#) [do\\_get\\_monthname](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const
- virtual [iter\\_type](#) [do\\_get\\_time](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const

- virtual [iter\\_type do\\_get\\_weekday](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const
- virtual [iter\\_type do\\_get\\_year](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const

### Static Protected Member Functions

- static [\\_\\_c\\_locale \\_S\\_clone\\_c\\_locale](#) ([\\_\\_c\\_locale](#) &\_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) ([\\_\\_c\\_locale](#) &\_\_cloc, const char \*\_\_s, [\\_\\_c\\_locale](#) \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) ([\\_\\_c\\_locale](#) &\_\_cloc)
- static [\\_\\_c\\_locale \\_S\\_get\\_c\\_locale](#) ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static [\\_\\_c\\_locale \\_S\\_lc\\_ctype\\_c\\_locale](#) ([\\_\\_c\\_locale](#) \_\_cloc, const char \*\_\_s)

### Friends

- class [locale::Impl](#)

### 5.708.1 Detailed Description

`template<typename _CharT, typename _InIter> class std::time_get< _CharT, _InIter >`

Primary class template [time\\_get](#).

This facet encapsulates the code to parse and return a date or time from a string. It is used by the istream numeric extraction operators. The [time\\_get](#) template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the [time\\_get](#) facet.

Definition at line 368 of file `locale_facets_nonio.h`.

### 5.708.2 Member Typedef Documentation

**5.708.2.1** `template<typename _CharT, typename _InIter> typedef _CharT std::time_get< _CharT, _InIter >::char_type`

Public typedefs.

Reimplemented in [std::time\\_get\\_byname< \\_CharT, \\_InIter >](#).

Definition at line 374 of file `locale_facets_nonio.h`.

### 5.708.2.2 `template<typename _CharT, typename _InIter> typedef _InIter std::time_get<_CharT, _InIter>::iter_type`

Public typedefs.

Reimplemented in `std::time_get_byname<_CharT, _InIter>`.

Definition at line 375 of file `locale_facets_nonio.h`.

## 5.708.3 Constructor & Destructor Documentation

### 5.708.3.1 `template<typename _CharT, typename _InIter> std::time_get< _CharT, _InIter>::time_get ( size_t __refs = 0 ) [inline, explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

#### Parameters

*refs* Passed to the base facet class.

Definition at line 390 of file `locale_facets_nonio.h`.

### 5.708.3.2 `template<typename _CharT, typename _InIter> virtual std::time_get<_CharT, _InIter>::~~time_get ( ) [inline, protected, virtual]`

Destructor.

Definition at line 546 of file `locale_facets_nonio.h`.

## 5.708.4 Member Function Documentation

### 5.708.4.1 `template<typename _CharT, typename _InIter> dateorder std::time_get<_CharT, _InIter>::date_order ( ) const [inline]`

Return preferred order of month, day, and year.

This function returns an enum from `timebase::dateorder` giving the preferred ordering if the format *x* given to `time_put::put()` only uses month, day, and year. If the format *x* for the associated locale uses other fields, this function returns `timebase::dateorder::noorder`.

NOTE: The library always returns `noorder` at the moment.

### Returns

A member of `timebase::dateorder`.

Definition at line 407 of file `locale_facets_nonio.h`.

References `std::time_get< _CharT, _InIter >::do_date_order()`.

**5.708.4.2** `template<typename _CharT , typename _InIter >  
time_base::dateorder std::time_get< _CharT, _InIter  
>::do_date_order ( ) const [protected, virtual]`

Return preferred order of month, day, and year.

This function returns an enum from `timebase::dateorder` giving the preferred ordering if the format *x* given to `time_put::put()` only uses month, day, and year. This function is a hook for derived classes to change the value returned.

### Returns

A member of `timebase::dateorder`.

Definition at line 620 of file `locale_facets_nonio.tcc`.

Referenced by `std::time_get< _CharT, _InIter >::date_order()`.

**5.708.4.3** `template<typename _CharT , typename _InIter > _InIter  
std::time_get< _CharT, _InIter >::do_get_date ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *  
__tm ) const [protected, virtual]`

Parse input date string.

This function parses a date according to the format *X* and puts the results into a user-supplied struct `tm`. This function is a hook for derived classes to change the value returned.

**See also**

[get\\_date\(\)](#) for details.

**Parameters**

*beg* Start of string to parse.  
*end* End of string to parse.  
*io* Source of the locale.  
*err* Error flags to set.  
*tm* Pointer to struct tm to fill in.

**Returns**

Iterator to first char beyond date string.

Definition at line 1047 of file locale\_facets\_nonio.tcc.

References std::ios\_base::\_M\_getloc(), and std::ios\_base::eofbit.

Referenced by std::time\_get< \_CharT, \_InIter >::get\_date().

**5.708.4.4** `template<typename _CharT, typename _InIter > _InIter  
std::time_get< _CharT, _InIter >::do_get_monthname ( iter_type  
__beg, iter_type __end, ios_base & __io, ios_base::iostate & __err,  
tm * __tm ) const [protected, virtual]`

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct tm.  
This function is a hook for derived classes to change the value returned.

**See also**

[get\\_monthname\(\)](#) for details.

**Parameters**

*beg* Start of string to parse.  
*end* End of string to parse.  
*io* Source of the locale.  
*err* Error flags to set.  
*tm* Pointer to struct tm to fill in.

**Returns**

Iterator to first char beyond month name.



Definition at line 1092 of file locale\_facets\_nonio.tcc.

References std::ios\_base::\_M\_getloc(), std::ios\_base::eofbit, std::ios\_base::failbit, and std::ios\_base::goodbit.

Referenced by std::time\_get< \_CharT, \_InIter >::get\_monthname().

**5.708.4.5** `template<typename _CharT, typename _InIter> _InIter  
std::time_get< _CharT, _InIter >::do_get_time ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *  
__tm ) const [protected, virtual]`

Parse input time string.

This function parses a time according to the format *x* and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

**See also**

[get\\_time\(\)](#) for details.

#### Parameters

*beg* Start of string to parse.  
*end* End of string to parse.  
*io* Source of the locale.  
*err* Error flags to set.  
*tm* Pointer to struct tm to fill in.

#### Returns

Iterator to first char beyond time string.

Definition at line 1030 of file locale\_facets\_nonio.tcc.

References std::ios\_base::\_M\_getloc(), and std::ios\_base::eofbit.

Referenced by std::time\_get< \_CharT, \_InIter >::get\_time().

**5.708.4.6** `template<typename _CharT, typename _InIter> _InIter  
std::time_get< _CharT, _InIter >::do_get_weekday ( iter_type  
__beg, iter_type __end, ios_base & __io, ios_base::iostate & __err,  
tm * __tm ) const [protected, virtual]`

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

#### See also

[get\\_weekday\(\)](#) for details.

#### Parameters

*beg* Start of string to parse.

*end* End of string to parse.

*io* Source of the locale.

*err* Error flags to set.

*tm* Pointer to struct tm to fill in.

#### Returns

Iterator to first char beyond weekday name.

Definition at line 1064 of file locale\_facets\_nonio.tcc.

References std::ios\_base::\_M\_getloc(), std::ios\_base::eofbit, std::ios\_base::failbit, and std::ios\_base::goodbit.

Referenced by std::time\_get< \_CharT, \_InIter >::get\_weekday().

```
5.708.4.7 template<typename _CharT, typename _InIter > _InIter
std::time_get< _CharT, _InIter >::do_get_year (iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *
__tm) const [protected, virtual]
```

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

#### See also

[get\\_year\(\)](#) for details.

#### Parameters

*beg* Start of string to parse.

*end* End of string to parse.

*io* Source of the locale.

*err* Error flags to set.

*tm* Pointer to struct tm to fill in.

### Returns

Iterator to first char beyond year.

Definition at line 1120 of file locale\_facets\_nonio.tcc.

References `std::ios_base::_M_getloc()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, and `std::ios_base::goodbit`.

Referenced by `std::time_get< _CharT, _InIter >::get_year()`.

**5.708.4.8** `template<typename _CharT, typename _InIter> iter_type  
std::time_get< _CharT, _InIter >::get_date ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *  
__tm ) const [inline]`

Parse input date string.

This function parses a date according to the format *x* and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_date()`.

If there is a valid date string according to format *x*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the date string. If an error occurs before the end, `err` |= `ios_base::failbit`. If parsing reads all the characters, `err` |= `ios_base::eofbit`.

### Parameters

*beg* Start of string to parse.

*end* End of string to parse.

*io* Source of the locale.

*err* Error flags to set.

*tm* Pointer to struct tm to fill in.

### Returns

Iterator to first char beyond date string.

Definition at line 456 of file locale\_facets\_nonio.h.

References `std::time_get< _CharT, _InIter >::do_get_date()`.

**5.708.4.9** `template<typename _CharT, typename _InIter > iter_type  
std::time_get< _CharT, _InIter >::get_monthname ( iter_type  
__beg, iter_type __end, ios_base & __io, ios_base::iostate & __err,  
tm * __tm ) const [inline]`

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct tm. The result is returned by calling [time\\_get::do\\_get\\_monthname\(\)](#).

Parsing starts by parsing an abbreviated month name. If a valid abbreviation is followed by a character that would lead to the full month name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, err |= [ios\\_base::failbit](#). If parsing reads all the characters, err |= [ios\\_base::eofbit](#).

#### Parameters

- beg* Start of string to parse.
- end* End of string to parse.
- io* Source of the locale.
- err* Error flags to set.
- tm* Pointer to struct tm to fill in.

#### Returns

Iterator to first char beyond month name.

Definition at line 513 of file locale\_facets\_nonio.h.

References [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_monthname\(\)](#).

**5.708.4.10** `template<typename _CharT, typename _InIter > iter_type  
std::time_get< _CharT, _InIter >::get_time ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *  
__tm ) const [inline]`

Parse input time string.

This function parses a time according to the format *X* and puts the results into a user-supplied struct tm. The result is returned by calling [time\\_get::do\\_get\\_time\(\)](#).

If there is a valid time string according to format *X*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the time string. If an error occurs before the end, err |= [ios\\_base::failbit](#). If parsing reads all the characters, err |= [ios\\_base::eofbit](#).

#### Parameters

*beg* Start of string to parse.  
*end* End of string to parse.  
*io* Source of the locale.  
*err* Error flags to set.  
*tm* Pointer to struct tm to fill in.

#### Returns

Iterator to first char beyond time string.

Definition at line 431 of file locale\_facets\_nonio.h.

References [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_time\(\)](#).

**5.708.4.11** `template<typename _CharT, typename _InIter> iter_type  
 std::time_get< _CharT, _InIter >::get_weekday ( iter_type __beg,  
 iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *  
 __tm ) const [inline]`

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. The result is returned by calling [time\\_get::do\\_get\\_weekday\(\)](#).

Parsing starts by parsing an abbreviated weekday name. If a valid abbreviation is followed by a character that would lead to the full weekday name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, err |= [ios\\_base::failbit](#). If parsing reads all the characters, err |= [ios\\_base::eofbit](#).

#### Parameters

*beg* Start of string to parse.  
*end* End of string to parse.  
*io* Source of the locale.  
*err* Error flags to set.

*tm* Pointer to struct tm to fill in.

### Returns

Iterator to first char beyond weekday name.

Definition at line 484 of file locale\_facets\_nonio.h.

References std::time\_get< \_CharT, \_InIter >::do\_get\_weekday().

```
5.708.4.12 template<typename _CharT, typename _InIter > iter_type
std::time_get< _CharT, _InIter >::get_year (iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *
__tm) const [inline]
```

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. The result is returned by calling [time\\_get::do\\_get\\_year\(\)](#).

4 consecutive digits are interpreted as a full year. If there are exactly 2 consecutive digits, the library interprets this as the number of years since 1900.

If an error occurs before the end, err |= [ios\\_base::failbit](#). If parsing reads all the characters, err |= [ios\\_base::eofbit](#).

### Parameters

*beg* Start of string to parse.

*end* End of string to parse.

*io* Source of the locale.

*err* Error flags to set.

*tm* Pointer to struct tm to fill in.

### Returns

Iterator to first char beyond year.

Definition at line 539 of file locale\_facets\_nonio.h.

References std::time\_get< \_CharT, \_InIter >::do\_get\_year().

## 5.709 `std::time_get_byname<_CharT, _InIter>` Class Template Reference 3308

### 5.708.5 Member Data Documentation

#### 5.708.5.1 `template<typename _CharT, typename _InIter> locale::id std::time_get<_CharT, _InIter>::id [static]`

Numpunct facet id.

Definition at line 380 of file `locale_facets_nonio.h`.

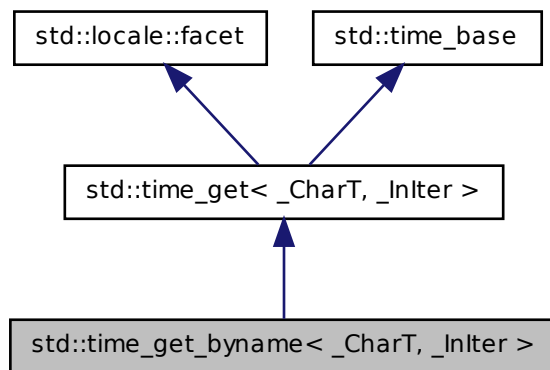
The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [locale\\_facets\\_nonio.tcc](#)

## 5.709 `std::time_get_byname<_CharT, _InIter>` Class Template Reference

class [time\\_get\\_byname](#) [22.2.5.2].

Inheritance diagram for `std::time_get_byname<_CharT, _InIter>`:



### Public Types

- typedef [basic\\_string](#)<\_CharT> `__string_type`

- typedef `_CharT` `char_type`
- enum `dateorder` {  
    `no_order`, `dmy`, `mdy`, `ymd`,  
    `ydm` }
- typedef `_InIter` `iter_type`

### Public Member Functions

- `time_get_byname` (const char \*, size\_t \_\_refs=0)
- dateorder `date_order` () const
- `iter_type` `get_date` (`iter_type` \_\_beg, `iter_type` \_\_end, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err, tm \*\_\_tm) const
- `iter_type` `get_monthname` (`iter_type` \_\_beg, `iter_type` \_\_end, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err, tm \*\_\_tm) const
- `iter_type` `get_time` (`iter_type` \_\_beg, `iter_type` \_\_end, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err, tm \*\_\_tm) const
- `iter_type` `get_weekday` (`iter_type` \_\_beg, `iter_type` \_\_end, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err, tm \*\_\_tm) const
- `iter_type` `get_year` (`iter_type` \_\_beg, `iter_type` \_\_end, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err, tm \*\_\_tm) const

### Static Public Attributes

- static `locale::id` `id`

### Protected Member Functions

- `iter_type` `_M_extract_name` (`iter_type` \_\_beg, `iter_type` \_\_end, int &\_\_member, const `_CharT` \*\_\_names, size\_t \_\_indexlen, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err) const
- `iter_type` `_M_extract_num` (`iter_type` \_\_beg, `iter_type` \_\_end, int &\_\_member, int \_\_min, int \_\_max, size\_t \_\_len, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err) const
- `iter_type` `_M_extract_via_format` (`iter_type` \_\_beg, `iter_type` \_\_end, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err, tm \*\_\_tm, const `_CharT` \*\_\_format) const
- `iter_type` `_M_extract_wday_or_month` (`iter_type` \_\_beg, `iter_type` \_\_end, int &\_\_member, const `_CharT` \*\_\_names, size\_t \_\_indexlen, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err) const
- virtual dateorder `do_date_order` () const
- virtual `iter_type` `do_get_date` (`iter_type` \_\_beg, `iter_type` \_\_end, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err, tm \*\_\_tm) const
- virtual `iter_type` `do_get_monthname` (`iter_type` \_\_beg, `iter_type` \_\_end, `ios_base` &\_\_io, `ios_base::iostate` &\_\_err, tm \*\_\_tm) const



## 5.709 std::time\_get\_byname<\_CharT, \_InIter> Class Template Reference 3310

- virtual [iter\\_type do\\_get\\_time](#) ([iter\\_type \\_\\_beg](#), [iter\\_type \\_\\_end](#), [ios\\_base &\\_\\_io](#), [ios\\_base::iostate &\\_\\_err](#), [tm \\*\\_\\_tm](#)) const
- virtual [iter\\_type do\\_get\\_weekday](#) ([iter\\_type \\_\\_beg](#), [iter\\_type \\_\\_end](#), [ios\\_base &\\_\\_io](#), [ios\\_base::iostate &\\_\\_err](#), [tm \\*\\_\\_tm](#)) const
- virtual [iter\\_type do\\_get\\_year](#) ([iter\\_type \\_\\_beg](#), [iter\\_type \\_\\_end](#), [ios\\_base &\\_\\_io](#), [ios\\_base::iostate &\\_\\_err](#), [tm \\*\\_\\_tm](#)) const

### Static Protected Member Functions

- static [\\_\\_c\\_locale \\_S\\_clone\\_c\\_locale](#) ([\\_\\_c\\_locale &\\_\\_cloc](#)) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) ([\\_\\_c\\_locale &\\_\\_cloc](#), const char \*\_\_s, [\\_\\_c\\_locale \\_\\_old=0](#))
- static void [\\_S\\_destroy\\_c\\_locale](#) ([\\_\\_c\\_locale &\\_\\_cloc](#))
- static [\\_\\_c\\_locale \\_S\\_get\\_c\\_locale](#) ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static [\\_\\_c\\_locale \\_S\\_lc\\_ctype\\_c\\_locale](#) ([\\_\\_c\\_locale \\_\\_cloc](#), const char \*\_\_s)

### Friends

- class [locale::Impl](#)

### 5.709.1 Detailed Description

**template<typename \_CharT, typename \_InIter> class std::time\_get\_byname<\_CharT, \_InIter>**

class [time\\_get\\_byname](#) [22.2.5.2].

Definition at line 686 of file [locale\\_facets\\_nonio.h](#).

### 5.709.2 Member Typedef Documentation

**5.709.2.1 template<typename \_CharT, typename \_InIter> typedef \_CharT std::time\_get\_byname<\_CharT, \_InIter>::char\_type**

Public typedefs.

Reimplemented from [std::time\\_get<\\_CharT, \\_InIter>](#).

Definition at line 690 of file [locale\\_facets\\_nonio.h](#).

## 5.709 std::time\_get\_byname<\_CharT, \_InIter> Class Template Reference 3311

### 5.709.2.2 template<typename \_CharT, typename \_InIter> typedef \_InIter std::time\_get\_byname<\_CharT, \_InIter>::iter\_type

Public typedefs.

Reimplemented from [std::time\\_get<\\_CharT, \\_InIter>](#).

Definition at line 691 of file locale\_facets\_nonio.h.

### 5.709.3 Member Function Documentation

#### 5.709.3.1 template<typename \_CharT, typename \_InIter> dateorder std::time\_get<\_CharT, \_InIter>::date\_order ( ) const [inline, inherited]

Return preferred order of month, day, and year.

This function returns an enum from [timebase::dateorder](#) giving the preferred ordering if the format *x* given to [time\\_put::put\(\)](#) only uses month, day, and year. If the format *x* for the associated locale uses other fields, this function returns [timebase::dateorder::noorder](#).

NOTE: The library always returns noorder at the moment.

#### Returns

A member of [timebase::dateorder](#).

Definition at line 407 of file locale\_facets\_nonio.h.

References [std::time\\_get<\\_CharT, \\_InIter>::do\\_date\\_order\(\)](#).

#### 5.709.3.2 template<typename \_CharT, typename \_InIter> time\_base::dateorder std::time\_get<\_CharT, \_InIter> >::do\_date\_order ( ) const [protected, virtual, inherited]

Return preferred order of month, day, and year.

This function returns an enum from [timebase::dateorder](#) giving the preferred ordering if the format *x* given to [time\\_put::put\(\)](#) only uses month, day, and year. This function is a hook for derived classes to change the value returned.

**Returns**

A member of timebase::dateorder.

Definition at line 620 of file locale\_facets\_nonio.tcc.

Referenced by std::time\_get<\_CharT, \_InIter>::date\_order().

**5.709.3.3** `template<typename _CharT, typename _InIter> _InIter  
std::time_get<_CharT, _InIter>::do_get_date ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *  
__tm ) const [protected, virtual, inherited]`

Parse input date string.

This function parses a date according to the format *X* and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

**See also**

[get\\_date\(\)](#) for details.

**Parameters**

*beg* Start of string to parse.  
*end* End of string to parse.  
*io* Source of the locale.  
*err* Error flags to set.  
*tm* Pointer to struct tm to fill in.

**Returns**

Iterator to first char beyond date string.

Definition at line 1047 of file locale\_facets\_nonio.tcc.

References std::ios\_base::\_M\_getloc(), and std::ios\_base::eofbit.

Referenced by std::time\_get<\_CharT, \_InIter>::get\_date().

**5.709.3.4** `template<typename _CharT, typename _InIter> _InIter  
std::time_get<_CharT, _InIter>::do_get_monthname ( iter_type  
__beg, iter_type __end, ios_base & __io, ios_base::iostate & __err,  
tm * __tm ) const [protected, virtual, inherited]`

### 5.709 std::time\_get\_byname<\_CharT, \_InIter> Class Template Reference 3313

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

#### See also

[get\\_monthname\(\)](#) for details.

#### Parameters

- beg* Start of string to parse.
- end* End of string to parse.
- io* Source of the locale.
- err* Error flags to set.
- tm* Pointer to struct tm to fill in.

#### Returns

Iterator to first char beyond month name.

Definition at line 1092 of file locale\_facets\_nonio.tcc.

References `std::ios_base::_M_getloc()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, and `std::ios_base::goodbit`.

Referenced by `std::time_get<_CharT, _InIter>::get_monthname()`.

```
5.709.3.5 template<typename _CharT, typename _InIter> _InIter
std::time_get<_CharT, _InIter>::do_get_time (iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *
__tm) const [protected, virtual, inherited]
```

Parse input time string.

This function parses a time according to the format *x* and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

#### See also

[get\\_time\(\)](#) for details.

#### Parameters

- beg* Start of string to parse.
- end* End of string to parse.

## 5.709 `std::time_get_byname<_CharT, _InIter>` Class Template Reference 3314

---

*io* Source of the locale.

*err* Error flags to set.

*tm* Pointer to struct tm to fill in.

### Returns

Iterator to first char beyond time string.

Definition at line 1030 of file locale\_facets\_nonio.tcc.

References `std::ios_base::_M_getloc()`, and `std::ios_base::eofbit`.

Referenced by `std::time_get<_CharT, _InIter>::get_time()`.

**5.709.3.6** `template<typename _CharT, typename _InIter> _InIter  
std::time_get<_CharT, _InIter>::do_get_weekday ( iter_type  
__beg, iter_type __end, ios_base & __io, ios_base::iostate & __err,  
tm * __tm ) const [protected, virtual, inherited]`

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

### See also

[get\\_weekday\(\)](#) for details.

### Parameters

*beg* Start of string to parse.

*end* End of string to parse.

*io* Source of the locale.

*err* Error flags to set.

*tm* Pointer to struct tm to fill in.

### Returns

Iterator to first char beyond weekday name.

Definition at line 1064 of file locale\_facets\_nonio.tcc.

References `std::ios_base::_M_getloc()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, and `std::ios_base::goodbit`.

Referenced by `std::time_get<_CharT, _InIter>::get_weekday()`.

**5.709.3.7** `template<typename _CharT, typename _InIter> _InIter  
std::time_get<_CharT, _InIter>::do_get_year ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *  
__tm ) const [protected, virtual, inherited]`

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

#### See also

[get\\_year\(\)](#) for details.

#### Parameters

*beg* Start of string to parse.  
*end* End of string to parse.  
*io* Source of the locale.  
*err* Error flags to set.  
*tm* Pointer to struct tm to fill in.

#### Returns

Iterator to first char beyond year.

Definition at line 1120 of file locale\_facets\_nonio.tcc.

References std::ios\_base::\_M\_getloc(), std::ios\_base::eofbit, std::ios\_base::failbit, and std::ios\_base::goodbit.

Referenced by std::time\_get<\_CharT, \_InIter>::get\_year().

**5.709.3.8** `template<typename _CharT, typename _InIter> iter_type  
std::time_get<_CharT, _InIter>::get_date ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *  
__tm ) const [inline, inherited]`

Parse input date string.

This function parses a date according to the format *x* and puts the results into a user-supplied struct tm. The result is returned by calling [time\\_get::do\\_get\\_date\(\)](#).

If there is a valid date string according to format *x*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the date string. If an error

## 5.709 std::time\_get\_byname<\_CharT, \_InIter > Class Template Reference 3316

occurs before the end, err |= [ios\\_base::failbit](#). If parsing reads all the characters, err |= [ios\\_base::eofbit](#).

### Parameters

- beg* Start of string to parse.
- end* End of string to parse.
- io* Source of the locale.
- err* Error flags to set.
- tm* Pointer to struct tm to fill in.

### Returns

Iterator to first char beyond date string.

Definition at line 456 of file locale\_facets\_nonio.h.

References [std::time\\_get<\\_CharT, \\_InIter >::do\\_get\\_date\(\)](#).

**5.709.3.9** `template<typename _CharT, typename _InIter > iter_type  
std::time_get<_CharT, _InIter >::get_monthname ( iter_type  
__beg, iter_type __end, ios_base & __io, ios_base::iostate & __err,  
tm * __tm ) const [inline, inherited]`

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct tm. The result is returned by calling [time\\_get::do\\_get\\_monthname\(\)](#).

Parsing starts by parsing an abbreviated month name. If a valid abbreviation is followed by a character that would lead to the full month name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, err |= [ios\\_base::failbit](#). If parsing reads all the characters, err |= [ios\\_base::eofbit](#).

### Parameters

- beg* Start of string to parse.
- end* End of string to parse.
- io* Source of the locale.
- err* Error flags to set.
- tm* Pointer to struct tm to fill in.

**Returns**

Iterator to first char beyond month name.

Definition at line 513 of file locale\_facets\_nonio.h.

References `std::time_get<_CharT, _InIter>::do_get_monthname()`.

**5.709.3.10** `template<typename _CharT, typename _InIter> iter_type  
std::time_get<_CharT, _InIter>::get_time ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *  
__tm ) const [inline, inherited]`

Parse input time string.

This function parses a time according to the format *X* and puts the results into a user-supplied struct *tm*. The result is returned by calling `time_get::do_get_time()`.

If there is a valid time string according to format *X*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the time string. If an error occurs before the end, `err` |= `ios_base::failbit`. If parsing reads all the characters, `err` |= `ios_base::eofbit`.

**Parameters**

*beg* Start of string to parse.

*end* End of string to parse.

*io* Source of the locale.

*err* Error flags to set.

*tm* Pointer to struct *tm* to fill in.

**Returns**

Iterator to first char beyond time string.

Definition at line 431 of file locale\_facets\_nonio.h.

References `std::time_get<_CharT, _InIter>::do_get_time()`.

**5.709.3.11** `template<typename _CharT, typename _InIter> iter_type  
std::time_get<_CharT, _InIter>::get_weekday ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *  
__tm ) const [inline, inherited]`



Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. The result is returned by calling [time\\_get::do\\_get\\_weekday\(\)](#).

Parsing starts by parsing an abbreviated weekday name. If a valid abbreviation is followed by a character that would lead to the full weekday name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, err |= [ios\\_base::failbit](#). If parsing reads all the characters, err |= [ios\\_base::eofbit](#).

#### Parameters

- beg* Start of string to parse.
- end* End of string to parse.
- io* Source of the locale.
- err* Error flags to set.
- tm* Pointer to struct tm to fill in.

#### Returns

Iterator to first char beyond weekday name.

Definition at line 484 of file locale\_facets\_nonio.h.

References [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_weekday\(\)](#).

**5.709.3.12** `template<typename _CharT, typename _InIter> iter_type  
std::time_get< _CharT, _InIter >::get_year ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *  
__tm ) const [inline, inherited]`

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. The result is returned by calling [time\\_get::do\\_get\\_year\(\)](#).

4 consecutive digits are interpreted as a full year. If there are exactly 2 consecutive digits, the library interprets this as the number of years since 1900.

If an error occurs before the end, err |= [ios\\_base::failbit](#). If parsing reads all the characters, err |= [ios\\_base::eofbit](#).

#### Parameters

- beg* Start of string to parse.

*end* End of string to parse.  
*io* Source of the locale.  
*err* Error flags to set.  
*tm* Pointer to struct tm to fill in.

#### Returns

Iterator to first char beyond year.

Definition at line 539 of file locale\_facets\_nonio.h.

References std::time\_get< \_CharT, \_InIter >::do\_get\_year().

#### 5.709.4 Member Data Documentation

##### 5.709.4.1 template<typename \_CharT, typename \_InIter > locale::id std::time\_get< \_CharT, \_InIter >::id [static, inherited]

Numpunct facet id.

Definition at line 380 of file locale\_facets\_nonio.h.

The documentation for this class was generated from the following file:

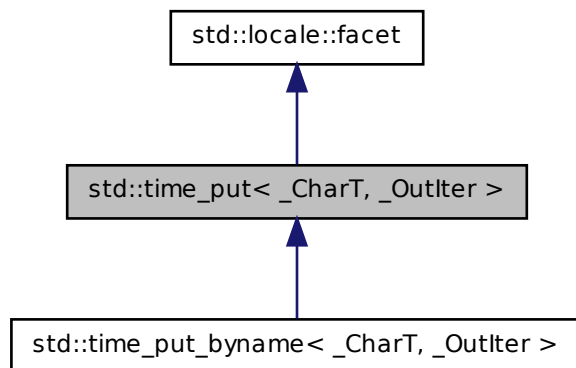
- [locale\\_facets\\_nonio.h](#)

#### 5.710 std::time\_put< \_CharT, \_OutIter > Class Template Reference

Primary class template [time\\_put](#).

This facet encapsulates the code to format and output dates and times according to formats used by strftime().

Inheritance diagram for `std::time_put<_CharT, _OutIter>`:



### Public Types

- typedef `_CharT` `char_type`
- typedef `_OutIter` `iter_type`

### Public Member Functions

- `time_put` (`size_t __refs=0`)
- `iter_type put` (`iter_type __s`, `ios_base &__io`, `char_type __fill`, `const tm *__tm`, `char __format`, `char __mod=0`) `const`
- `iter_type put` (`iter_type __s`, `ios_base &__io`, `char_type __fill`, `const tm *__tm`, `const _CharT *__beg`, `const _CharT *__end`) `const`

### Static Public Attributes

- static `locale::id` `id`

### Protected Member Functions

- virtual `~time_put` ()
- virtual `iter_type do_put` (`iter_type __s`, `ios_base &__io`, `char_type __fill`, `const tm *__tm`, `char __format`, `char __mod`) `const`

### Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

### Friends

- class **locale::\_Impl**

#### 5.710.1 Detailed Description

**template<typename \_CharT, typename \_OutIter> class std::time\_put< \_CharT, \_OutIter >**

Primary class template [time\\_put](#).

This facet encapsulates the code to format and output dates and times according to formats used by strftime(). The [time\\_put](#) template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the [time\\_put](#) facet.

Definition at line 715 of file locale\_facets\_nonio.h.

#### 5.710.2 Member Typedef Documentation

**5.710.2.1    template<typename \_CharT , typename \_OutIter > typedef \_CharT  
std::time\_put< \_CharT, \_OutIter >::char\_type**

Public typedefs.

Reimplemented in [std::time\\_put\\_byname< \\_CharT, \\_OutIter >](#).

Definition at line 721 of file locale\_facets\_nonio.h.

**5.710.2.2    template<typename \_CharT , typename \_OutIter > typedef \_OutIter  
std::time\_put< \_CharT, \_OutIter >::iter\_type**

Public typedefs.

Reimplemented in `std::time_put_byname<_CharT, _OutIter>`.

Definition at line 722 of file `locale_facets_nonio.h`.

### 5.710.3 Constructor & Destructor Documentation

**5.710.3.1** `template<typename _CharT, typename _OutIter> std::time_put<_CharT, _OutIter>::time_put ( size_t __refs = 0 ) [inline, explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

#### Parameters

*refs* Passed to the base facet class.

Definition at line 736 of file `locale_facets_nonio.h`.

**5.710.3.2** `template<typename _CharT, typename _OutIter> virtual std::time_put<_CharT, _OutIter>::~~time_put ( ) [inline, protected, virtual]`

Destructor.

Definition at line 782 of file `locale_facets_nonio.h`.

### 5.710.4 Member Function Documentation

**5.710.4.1** `template<typename _CharT, typename _OutIter> _OutIter std::time_put<_CharT, _OutIter>::do_put ( iter_type __s, ios_base & __io, char_type __fill, const tm * __tm, char __format, char __mod ) const [protected, virtual]`

Format and output a time or date.

This function formats the data in struct `tm` according to the provided format char and optional modifier. This function is a hook for derived classes to change the value returned.

**See also**

[put\(\)](#) for more details.

**Parameters**

*s* The stream to write to.  
*io* Source of locale.  
*fill* char\_type to use for padding.  
*tm* Struct tm with date and time info to format.  
*format* Format char.  
*mod* Optional modifier char.

**Returns**

Iterator after writing.

Definition at line 1178 of file locale\_facets\_nonio.tcc.

References `std::ios_base::_M_getloc()`, and `std::__ctype_abstract_base< _CharT >::widen()`.

Referenced by `std::time_put< _CharT, _OutIter >::put()`.

**5.710.4.2** `template<typename _CharT, typename _OutIter > _OutIter  
std::time_put< _CharT, _OutIter >::put ( iter_type __s, ios_base &  
__io, char_type __fill, const tm * __tm, const _CharT * __beg,  
const _CharT * __end ) const`

Format and output a time or date.

This function formats the data in struct tm according to the provided format string. The format string is interpreted as by `strftime()`.

**Parameters**

*s* The stream to write to.  
*io* Source of locale.  
*fill* char\_type to use for padding.  
*tm* Struct tm with date and time info to format.  
*beg* Start of format string.  
*end* End of format string.

### Returns

Iterator after writing.

Definition at line 1143 of file locale\_facets\_nonio.tcc.

References `std::ios_base::_M_getloc()`, `std::time_put<_CharT, _OutIter>::do_put()`, and `std::__ctype_abstract_base<_CharT>::narrow()`.

**5.710.4.3** `template<typename _CharT, typename _OutIter> iter_type  
std::time_put<_CharT, _OutIter>::put ( iter_type __s, ios_base  
& __io, char_type __fill, const tm * __tm, char __format, char  
__mod = 0 ) const [inline]`

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. The format and modifier are interpreted as by `strftime()`. It does so by returning `time_put::do_put()`.

### Parameters

- s* The stream to write to.
- io* Source of locale.
- fill* `char_type` to use for padding.
- tm* Struct tm with date and time info to format.
- format* Format char.
- mod* Optional modifier char.

### Returns

Iterator after writing.

Definition at line 775 of file locale\_facets\_nonio.h.

References `std::time_put<_CharT, _OutIter>::do_put()`.

## 5.710.5 Member Data Documentation

**5.710.5.1** `template<typename _CharT, typename _OutIter> locale::id  
std::time_put<_CharT, _OutIter>::id [static]`

Numpunct facet id.

## 5.711 `std::time_put_byname< _CharT, _OutIter >` Class Template Reference 3325

Definition at line 726 of file `locale_facets_nonio.h`.

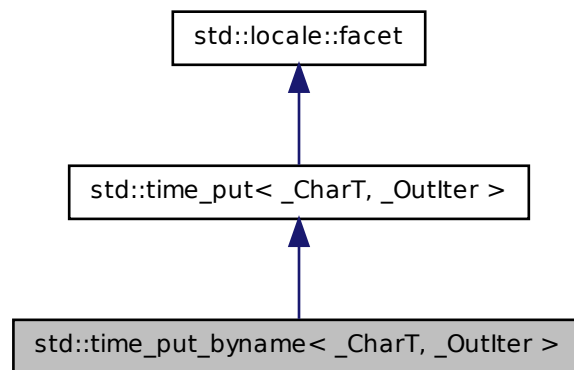
The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [locale\\_facets\\_nonio.tcc](#)

### 5.711 `std::time_put_byname< _CharT, _OutIter >` Class Template Reference

class [time\\_put\\_byname](#) [22.2.5.4].

Inheritance diagram for `std::time_put_byname< _CharT, _OutIter >`:



#### Public Types

- typedef `_CharT` [char\\_type](#)
- typedef `_OutIter` [iter\\_type](#)

#### Public Member Functions

- **time\_put\_byname** (`const char *`, `size_t __refs=0`)
- [iter\\_type put](#) ([iter\\_type](#) \_\_s, [ios\\_base](#) &\_\_io, [char\\_type](#) \_\_fill, `const tm *__tm`, `char __format`, `char __mod=0`) `const`



## 5.711 `std::time_put_byname<_CharT, _OutIter>` Class Template Reference

- `iter_type put (iter_type __s, ios_base &__io, char_type __fill, const tm *__tm, const _CharT *__beg, const _CharT *__end) const`

### Static Public Attributes

- static `locale::id id`

### Protected Member Functions

- virtual `iter_type do_put (iter_type __s, ios_base &__io, char_type __fill, const tm *__tm, char __format, char __mod) const`

### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static `void _S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static `void _S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static `const char * _S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

### Friends

- class `locale::_Impl`

#### 5.711.1 Detailed Description

`template<typename _CharT, typename _OutIter> class std::time_put_byname<_CharT, _OutIter>`

class `time_put_byname` [22.2.5.4].

Definition at line 811 of file `locale_facets_nonio.h`.

#### 5.711.2 Member Typedef Documentation

**5.711.2.1** `template<typename _CharT, typename _OutIter> typedef _CharT std::time_put_byname<_CharT, _OutIter>::char_type`

Public typedefs.

## 5.711 `std::time_put_byname<_CharT, _OutIter>` Class Template Reference 3327

Reimplemented from `std::time_put<_CharT, _OutIter>`.

Definition at line 815 of file `locale_facets_nonio.h`.

### 5.711.2.2 `template<typename _CharT, typename _OutIter> typedef _OutIter std::time_put_byname<_CharT, _OutIter>::iter_type`

Public typedefs.

Reimplemented from `std::time_put<_CharT, _OutIter>`.

Definition at line 816 of file `locale_facets_nonio.h`.

### 5.711.3 Member Function Documentation

#### 5.711.3.1 `template<typename _CharT, typename _OutIter> _OutIter std::time_put<_CharT, _OutIter>::do_put ( iter_type __s, ios_base & __io, char_type __fill, const tm * __tm, char __format, char __mod ) const` `[protected, virtual, inherited]`

Format and output a time or date.

This function formats the data in struct `tm` according to the provided format char and optional modifier. This function is a hook for derived classes to change the value returned.

#### See also

`put()` for more details.

#### Parameters

- s* The stream to write to.
- io* Source of locale.
- fill* `char_type` to use for padding.
- tm* Struct `tm` with date and time info to format.
- format* Format char.
- mod* Optional modifier char.

#### Returns

Iterator after writing.

## 5.711 std::time\_put\_byname<\_CharT, \_OutIter> Class Template Referenced 3328

Definition at line 1178 of file locale\_facets\_nonio.tcc.

References std::ios\_base::\_M\_getloc(), and std::\_\_ctype\_abstract\_base<\_CharT>::widen().

Referenced by std::time\_put<\_CharT, \_OutIter>::put().

**5.711.3.2** `template<typename _CharT, typename _OutIter> _OutIter  
std::time_put<_CharT, _OutIter>::put ( iter_type __s, ios_base &  
__io, char_type __fill, const tm * __tm, const _CharT * __beg,  
const _CharT * __end ) const [inherited]`

Format and output a time or date.

This function formats the data in struct tm according to the provided format string. The format string is interpreted as by strftime().

### Parameters

- s* The stream to write to.
- io* Source of locale.
- fill* char\_type to use for padding.
- tm* Struct tm with date and time info to format.
- beg* Start of format string.
- end* End of format string.

### Returns

- Iterator after writing.

Definition at line 1143 of file locale\_facets\_nonio.tcc.

References std::ios\_base::\_M\_getloc(), std::time\_put<\_CharT, \_OutIter>::do\_put(), and std::\_\_ctype\_abstract\_base<\_CharT>::narrow().

**5.711.3.3** `template<typename _CharT, typename _OutIter> iter_type  
std::time_put<_CharT, _OutIter>::put ( iter_type __s, ios_base  
& __io, char_type __fill, const tm * __tm, char __format, char  
__mod = 0 ) const [inline, inherited]`

Format and output a time or date.

This function formats the data in struct `tm` according to the provided format char and optional modifier. The format and modifier are interpreted as by `strftime()`. It does so by returning `time_put::do_put()`.

### Parameters

- s* The stream to write to.
- io* Source of locale.
- fill* `char_type` to use for padding.
- tm* Struct `tm` with date and time info to format.
- format* Format char.
- mod* Optional modifier char.

### Returns

Iterator after writing.

Definition at line 775 of file `locale_facets_nonio.h`.

References `std::time_put<_CharT, _OutIter>::do_put()`.

## 5.711.4 Member Data Documentation

### 5.711.4.1 `template<typename _CharT, typename _OutIter> locale::id` `std::time_put<_CharT, _OutIter>::id` [`static`, `inherited`]

Numpunct facet id.

Definition at line 726 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

## 5.712 `std::timed_mutex` Class Reference

[timed\\_mutex](#)

### Public Types

- `typedef __native_type * native_handle_type`

### Public Member Functions

- **timed\_mutex** (const [timed\\_mutex](#) &)
- void **lock** ()
- native\_handle\_type **native\_handle** ()
- [timed\\_mutex](#) & **operator=** (const [timed\\_mutex](#) &)
- bool **try\_lock** ()
- template<class \_Rep, class \_Period >  
bool **try\_lock\_for** (const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime)
- template<class \_Clock, class \_Duration >  
bool **try\_lock\_until** (const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_-  
atime)
- void **unlock** ()

#### 5.712.1 Detailed Description

[timed\\_mutex](#)

Definition at line 210 of file mutex.

The documentation for this class was generated from the following file:

- [mutex](#)

### 5.713 `std::try_to_lock_t` Struct Reference

Try to acquire ownership of the mutex without blocking.

#### 5.713.1 Detailed Description

Try to acquire ownership of the mutex without blocking.

Definition at line 425 of file mutex.

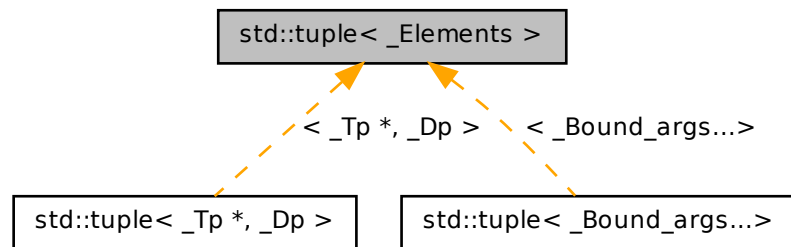
The documentation for this struct was generated from the following file:

- [mutex](#)

### 5.714 `std::tuple<_Elements>` Class Template Reference

tuple

Inheritance diagram for std::tuple< \_Elements >:



### Public Member Functions

- constexpr **tuple** (const \_Elements &...\_\_elements)
- constexpr **tuple** (const **tuple** &)
- **tuple** (**tuple** &&\_\_in)
- template<typename... \_UElements, typename = typename std::enable\_if<sizeof...(\_UElements) == sizeof...(\_Elements)>::type>  
**tuple** (\_UElements &&...\_\_elements)
- template<typename... \_UElements, typename = typename std::enable\_if<sizeof...(\_UElements) == sizeof...(\_Elements)>::type>  
**tuple** (const **tuple**< \_UElements...> &\_\_in)
- template<typename... \_UElements, typename = typename std::enable\_if<sizeof...(\_UElements) == sizeof...(\_Elements)>::type>  
**tuple** (**tuple**< \_UElements...> &&\_\_in)
- template<typename... \_UElements, typename = typename std::enable\_if<sizeof...(\_UElements) == sizeof...(\_Elements)>::type>  
**tuple** & **operator=** (const **tuple**< \_UElements...> &\_\_in)
- template<typename... \_UElements, typename = typename std::enable\_if<sizeof...(\_UElements) == sizeof...(\_Elements)>::type>  
**tuple** & **operator=** (**tuple**< \_UElements...> &&\_\_in)
- **tuple** & **operator=** (const **tuple** &\_\_in)
- **tuple** & **operator=** (**tuple** &&\_\_in)
- void **swap** (**tuple** &\_\_in)

**5.714.1 Detailed Description**

**template<typename... \_Elements> class std::tuple<\_Elements>**

tuple

Definition at line 230 of file tuple.

The documentation for this class was generated from the following file:

- [tuple](#)

**5.715 `std::tuple<_T1>` Class Template Reference**

tuple (1-element).

Inherits `_Tuple_impl<0, _T1>`.

**Public Member Functions**

- constexpr **tuple** (const \_T1 &\_\_a1)
- constexpr **tuple** (const [tuple](#) &)
- **tuple** ([tuple](#) &&\_\_in)
- template<typename \_U1 , typename = typename std::enable\_if<std::is\_convertible<\_U1, \_T1>::value>::type>  
**tuple** (\_U1 &&\_\_a1)
- template<typename \_U1 >  
**tuple** (const [tuple](#)<\_U1> &\_\_in)
- template<typename \_U1 >  
**tuple** ([tuple](#)<\_U1> &&\_\_in)
- template<typename \_U1 >  
[tuple](#) & **operator=** (const [tuple](#)<\_U1> &\_\_in)
- template<typename \_U1 >  
[tuple](#) & **operator=** ([tuple](#)<\_U1> &&\_\_in)
- [tuple](#) & **operator=** (const [tuple](#) &\_\_in)
- [tuple](#) & **operator=** ([tuple](#) &&\_\_in)
- void **swap** ([tuple](#) &\_\_in)

**5.715.1 Detailed Description**

**template<typename \_T1> class std::tuple<\_T1>**

tuple (1-element).

Definition at line 413 of file tuple.

The documentation for this class was generated from the following file:

- [tuple](#)

## 5.716 `std::tuple< _T1, _T2 >` Class Template Reference

tuple (2-element), with construction and assignment from a pair.

Inherits `_Tuple_impl< 0, _T1, _T2 >`.

### Public Member Functions

- constexpr **tuple** (const \_T1 &\_\_a1, const \_T2 &\_\_a2)
- constexpr **tuple** (const [tuple](#) &)
- template<typename \_U1, typename \_U2 >  
**tuple** (const [pair](#)< \_U1, \_U2 > &\_\_in)
- template<typename \_U1, typename \_U2 >  
**tuple** ([pair](#)< \_U1, \_U2 > &&\_\_in)
- **tuple** ([tuple](#) &&\_\_in)
- template<typename \_U1, typename \_U2 >  
**tuple** (\_U1 &&\_\_a1, \_U2 &&\_\_a2)
- template<typename \_U1, typename \_U2 >  
**tuple** (const [tuple](#)< \_U1, \_U2 > &\_\_in)
- template<typename \_U1, typename \_U2 >  
**tuple** ([tuple](#)< \_U1, \_U2 > &&\_\_in)
- template<typename \_U1, typename \_U2 >  
[tuple](#) & **operator=** (const [tuple](#)< \_U1, \_U2 > &\_\_in)
- [tuple](#) & **operator=** (const [tuple](#) &\_\_in)
- [tuple](#) & **operator=** ([tuple](#) &&\_\_in)
- template<typename \_U1, typename \_U2 >  
[tuple](#) & **operator=** ([tuple](#)< \_U1, \_U2 > &&\_\_in)
- template<typename \_U1, typename \_U2 >  
[tuple](#) & **operator=** (const [pair](#)< \_U1, \_U2 > &\_\_in)
- template<typename \_U1, typename \_U2 >  
[tuple](#) & **operator=** ([pair](#)< \_U1, \_U2 > &&\_\_in)
- void **swap** ([tuple](#) &\_\_in)

### 5.716.1 Detailed Description

**template<typename \_T1, typename \_T2> class std::tuple< \_T1, \_T2 >**

tuple (2-element), with construction and assignment from a pair.



Definition at line 315 of file tuple.

The documentation for this class was generated from the following file:

- [tuple](#)

## **5.717 `std::tuple_element< 0, tuple< _Head, _Tail...> >` Struct Template Reference**

### **Public Types**

- `typedef _Head type`

#### **5.717.1 Detailed Description**

`template<typename _Head, typename... _Tail> struct std::tuple_element< 0, tuple< _Head, _Tail...> >`

Basis case for `tuple_element`: The first element is the one we're seeking.

Definition at line 496 of file tuple.

The documentation for this struct was generated from the following file:

- [tuple](#)

## **5.718 `std::tuple_element< __i, tuple< _Head, _Tail...> >` Struct Template Reference**

Inherits `tuple_element< __i-1, tuple< _Tail...> >`.

#### **5.718.1 Detailed Description**

`template<std::size_t __i, typename _Head, typename... _Tail> struct std::tuple_element< __i, tuple< _Head, _Tail...> >`

Recursive case for `tuple_element`: strip off the first element in the tuple and retrieve the (i-1)th element of the remaining tuple.

Definition at line 489 of file tuple.

The documentation for this struct was generated from the following file:

- [tuple](#)

## **5.719 std::tuple\_size< tuple< \_Elements...> > Struct Template Reference**

class tuple\_size

### **Static Public Attributes**

- static const std::size\_t value

### **5.719.1 Detailed Description**

```
template<typename... _Elements> struct std::tuple_size< tuple< _Elements...>
>
```

class tuple\_size

Definition at line 507 of file tuple.

The documentation for this struct was generated from the following file:

- [tuple](#)

## **5.720 std::type\_index Struct Reference**

The class [type\\_index](#) provides a simple wrapper for [type\\_info](#) which can be used as an index type in associative containers (23.6) and in unordered associative containers (23.7).

### **Public Member Functions**

- **type\_index** (const [type\\_info](#) &\_\_rhs)
- size\_t **hash\_code** () const
- const char \* **name** () const
- bool **operator!=** (const [type\\_index](#) &\_\_rhs) const
- bool **operator<** (const [type\\_index](#) &\_\_rhs) const
- bool **operator<=** (const [type\\_index](#) &\_\_rhs) const
- bool **operator==** (const [type\\_index](#) &\_\_rhs) const
- bool **operator>** (const [type\\_index](#) &\_\_rhs) const
- bool **operator>=** (const [type\\_index](#) &\_\_rhs) const

### 5.720.1 Detailed Description

The class [type\\_index](#) provides a simple wrapper for [type\\_info](#) which can be used as an index type in associative containers (23.6) and in unordered associative containers (23.7).

Definition at line 49 of file `typeidindex`.

The documentation for this struct was generated from the following file:

- [typeidindex](#)

## 5.721 std::type\_info Class Reference

Part of RTTI.

Inherited by `__cxxabiv1::__array_type_info`, `__cxxabiv1::__class_type_info`, `__cxxabiv1::__enum_type_info`, `__cxxabiv1::__function_type_info`, `__cxxabiv1::__fundamental_type_info`, and `__cxxabiv1::__pbase_type_info`.

### Public Member Functions

- virtual [~type\\_info](#) ()
- virtual bool `__do_catch` (const [type\\_info](#) \* \_\_thr\_type, void \*\* \_\_thr\_obj, unsigned \_\_outer) const
- virtual bool `__do_upcast` (const `__cxxabiv1::__class_type_info` \* \_\_target, void \*\* \_\_obj\_ptr) const
- virtual bool `__is_function_p` () const
- virtual bool `__is_pointer_p` () const
- bool `before` (const [type\\_info](#) & \_\_arg) const
- `size_t hash_code` () const throw ()
- const char \* `name` () const
- bool `operator!=` (const [type\\_info](#) & \_\_arg) const
- bool `operator==` (const [type\\_info](#) & \_\_arg) const

### Protected Member Functions

- `type_info` (const char \* \_\_n)

### Protected Attributes

- const char \* `__name`

### 5.721.1 Detailed Description

Part of RTTI. The [type\\_info](#) class describes type information generated by an implementation.

Definition at line 91 of file typeinfo.

### 5.721.2 Constructor & Destructor Documentation

#### 5.721.2.1 virtual std::type\_info::~~type\_info ( ) [virtual]

Destructor first. Being the first non-inline virtual function, this controls in which translation unit the vtable is emitted. The compiler makes use of that information to know where to emit the runtime-mandated [type\\_info](#) structures in the new-abi.

### 5.721.3 Member Function Documentation

#### 5.721.3.1 const char\* std::type\_info::name ( ) const [inline]

Returns an *implementation-defined* byte string; this is not portable between compilers!

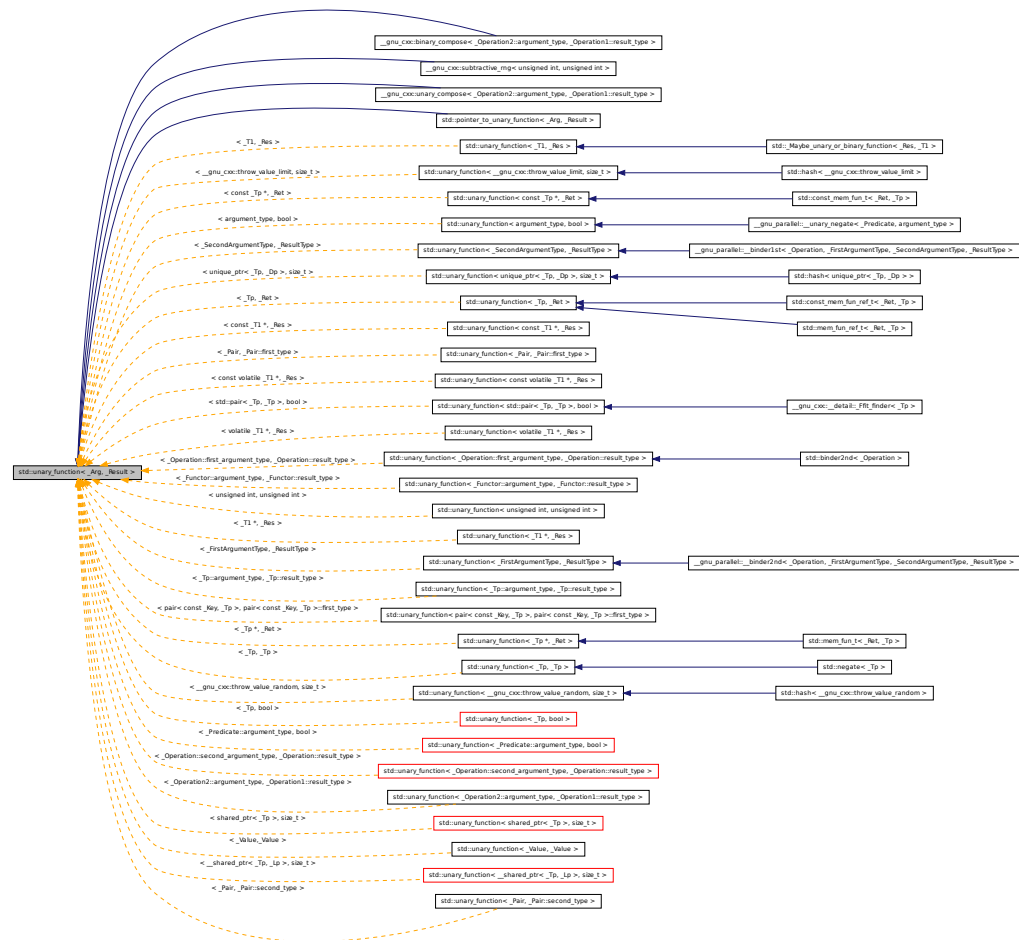
Definition at line 102 of file typeinfo.

The documentation for this class was generated from the following file:

- [typeinfo](#)

## 5.722 std::unary\_function< \_Arg, \_Result > Struct Template Reference

Inheritance diagram for std::unary\_function< \_Arg, \_Result >:



### Public Types

- typedef `_Arg` `argument_type`
- typedef `_Result` `result_type`

### 5.722.1 Detailed Description

**template<typename \_Arg, typename \_Result> struct std::unary\_function< \_Arg, \_Result >**

This is one of the [functor base classes](#).

Definition at line 102 of file `stl_function.h`.

### 5.722.2 Member Typedef Documentation

**5.722.2.1   template<typename \_Arg, typename \_Result> typedef \_Arg  
std::unary\_function< \_Arg, \_Result >::argument\_type**

`argument_type` is the type of the argument

Definition at line 105 of file `stl_function.h`.

**5.722.2.2   template<typename \_Arg, typename \_Result> typedef \_Result  
std::unary\_function< \_Arg, \_Result >::result\_type**

`result_type` is the return type

Definition at line 108 of file `stl_function.h`.

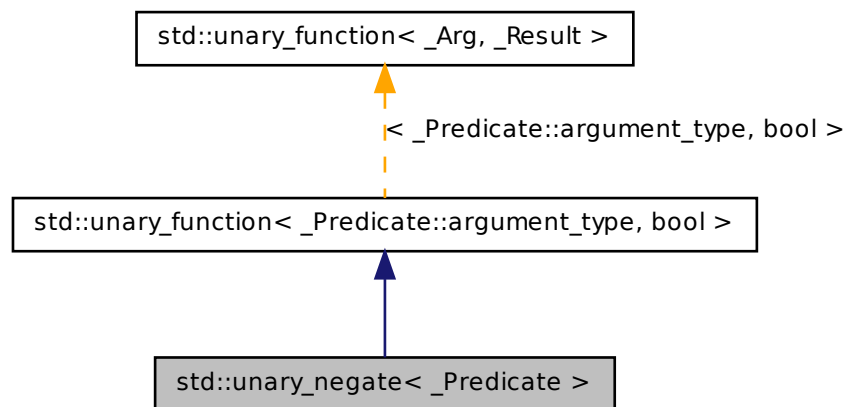
The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 5.723 `std::unary_negate<_Predicate>` Class Template Reference

One of the [negation functors](#).

Inheritance diagram for std::unary\_negate< \_Predicate >:



### Public Types

- typedef `_Predicate::argument_type` [argument\\_type](#)
- typedef `bool` [result\\_type](#)

### Public Member Functions

- **unary\_negate** (`const _Predicate &__x`)
- **operator()** (`const typename _Predicate::argument_type &__x`) `const`

### Protected Attributes

- `_Predicate _M_pred`

#### 5.723.1 Detailed Description

**template<typename \_Predicate> class std::unary\_negate< \_Predicate >**

One of the [negation functors](#).

Definition at line 352 of file `stl_function.h`.

### 5.723.2 Member Typedef Documentation

**5.723.2.1** `typedef _Predicate::argument_type std::unary_function<  
_Predicate::argument_type , bool >::argument_type  
[inherited]`

`argument_type` is the type of the argument

Definition at line 105 of file `stl_function.h`.

**5.723.2.2** `typedef bool std::unary_function< _Predicate::argument_type , bool  
>::result_type [inherited]`

`result_type` is the return type

Definition at line 108 of file `stl_function.h`.

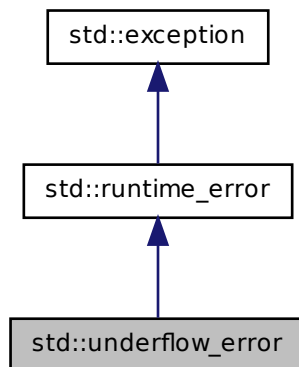
The documentation for this class was generated from the following file:

- [stl\\_function.h](#)



## 5.724 `std::underflow_error` Class Reference

Inheritance diagram for `std::underflow_error`:



### Public Member Functions

- **`underflow_error`** (const [string](#) &\_\_arg)
- virtual const char \* **`what`** () const throw ()

#### 5.724.1 Detailed Description

Thrown to indicate arithmetic underflow.

Definition at line 143 of file `stdexcept`.

#### 5.724.2 Member Function Documentation

##### 5.724.2.1 virtual const char\* `std::runtime_error::what` ( ) const throw () [virtual, inherited]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

## 5.725 `std::uniform_int_distribution< _IntType >` Class Template Reference 3343

The documentation for this class was generated from the following file:

- [stdexcept](#)

### 5.725 `std::uniform_int_distribution< _IntType >` Class Template Reference

Uniform discrete distribution for random numbers. A discrete random distribution on the range  $[min, max]$  with equal probability throughout the range.

#### Classes

- struct [param\\_type](#)

#### Public Types

- typedef `_IntType` [result\\_type](#)

#### Public Member Functions

- [uniform\\_int\\_distribution](#) (`_IntType __a=0, _IntType __b=std::numeric_limits<_IntType >::max()`)
- [uniform\\_int\\_distribution](#) (const [param\\_type](#) &\_\_p)
- [result\\_type a](#) () const
- [result\\_type b](#) () const
- [result\\_type max](#) () const
- [result\\_type min](#) () const
- template<typename `_UniformRandomNumberGenerator` >  
[result\\_type operator\(\)](#) (`_UniformRandomNumberGenerator &__urng, const param\_type &__p`)
- template<typename `_UniformRandomNumberGenerator` >  
[result\\_type operator\(\)](#) (`_UniformRandomNumberGenerator &__urng`)
- void [param](#) (const [param\\_type](#) &\_\_param)
- [param\\_type param](#) () const
- void [reset](#) ()

#### Public Attributes

- [param\\_type \\_M\\_param](#)

### 5.725.1 Detailed Description

**template<typename \_IntType = int> class std::uniform\_int\_distribution< \_IntType >**

Uniform discrete distribution for random numbers. A discrete random distribution on the range  $[min, max]$  with equal probability throughout the range.

Definition at line 1605 of file random.h.

### 5.725.2 Member Typedef Documentation

**5.725.2.1** `template<typename _IntType = int> typedef _IntType  
std::uniform_int_distribution< _IntType >::result_type`

The type of the range of the distribution.

Definition at line 1612 of file random.h.

### 5.725.3 Constructor & Destructor Documentation

**5.725.3.1** `template<typename _IntType = int> std::uniform_int_distribution<  
_IntType >::uniform_int_distribution ( _IntType __a = 0, _IntType  
__b = std::numeric_limits<_IntType>::max() ) [inline,  
explicit]`

Constructs a uniform distribution object.

Definition at line 1648 of file random.h.

### 5.725.4 Member Function Documentation

**5.725.4.1** `template<typename _IntType = int> result_type  
std::uniform_int_distribution< _IntType >::max ( ) const  
[inline]`

Returns the inclusive upper bound of the distribution range.

Definition at line 1700 of file random.h.

## 5.725 `std::uniform_int_distribution<_IntType>` Class Template Reference 3345

**5.725.4.2** `template<typename _IntType = int> result_type  
std::uniform_int_distribution<_IntType>::min ( ) const  
[inline]`

Returns the inclusive lower bound of the distribution range.

Definition at line 1693 of file random.h.

**5.725.4.3** `template<typename _IntType = int> template<typename  
_UniformRandomNumberGenerator> result_type  
std::uniform_int_distribution<_IntType>::operator() (   
_UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 1708 of file random.h.

References `std::uniform_int_distribution<_IntType>::operator()()`, and  
`std::uniform_int_distribution<_IntType>::param()`.

Referenced by `std::uniform_int_distribution<_IntType>::operator()()`.

**5.725.4.4** `template<typename _IntType = int> void std::uniform_int_  
distribution<_IntType>::param ( const param_type & __param )  
[inline]`

Sets the parameter set of the distribution.

### Parameters

**`__param`** The new parameter set of the distribution.

Definition at line 1686 of file random.h.

**5.725.4.5** `template<typename _IntType = int> param_type  
std::uniform_int_distribution<_IntType>::param ( ) const  
[inline]`

Returns the parameter set of the distribution.

Definition at line 1678 of file random.h.

## 5.726 `std::uniform_int_distribution< _IntType >::param_type` Struct Reference 3346

---

Referenced by `std::uniform_int_distribution< _IntType >::operator()()`, `std::operator==( )`, and `std::operator>>()`.

**5.725.4.6** `template<typename _IntType = int> void  
std::uniform_int_distribution< _IntType >::reset ( ) [inline]`

Resets the distribution state.

Does nothing for the uniform integer distribution.

Definition at line 1664 of file `random.h`.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.726 `std::uniform_int_distribution< _IntType >::param_type` Struct Reference

### Public Types

- typedef [uniform\\_int\\_distribution< \\_IntType >](#) **distribution\_type**

### Public Member Functions

- **param\_type** (`_IntType __a=0, _IntType __b=std::numeric_limits< _IntType >::max()`)
- [result\\_type a](#) () const
- [result\\_type b](#) () const

### Friends

- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

### 5.726.1 Detailed Description

`template<typename _IntType = int> struct std::uniform_int_distribution< _IntType >::param_type`

Parameter type.

Definition at line 1614 of file `random.h`.

## 5.727 `std::uniform_real_distribution<_RealType>` Class Template Reference 3347

The documentation for this struct was generated from the following file:

- [random.h](#)

### 5.727 `std::uniform_real_distribution<_RealType>` Class Template Reference

Uniform continuous distribution for random numbers.

#### Classes

- struct [param\\_type](#)

#### Public Types

- typedef `_RealType` [result\\_type](#)

#### Public Member Functions

- [uniform\\_real\\_distribution](#) (`_RealType __a=_RealType(0), _RealType __b=_RealType(1)`)
- [uniform\\_real\\_distribution](#) (const [param\\_type](#) &\_\_p)
- [result\\_type a](#) () const
- [result\\_type b](#) () const
- [result\\_type max](#) () const
- [result\\_type min](#) () const
- template<typename `_UniformRandomNumberGenerator`> [result\\_type operator\(\)](#) (`_UniformRandomNumberGenerator &__urng, const param\_type &__p`)
- template<typename `_UniformRandomNumberGenerator`> [result\\_type operator\(\)](#) (`_UniformRandomNumberGenerator &__urng`)
- void [param](#) (const [param\\_type](#) &\_\_param)
- [param\\_type param](#) () const
- void [reset](#) ()

#### 5.727.1 Detailed Description

`template<typename _RealType = double> class std::uniform_real_distribution<_RealType>`

Uniform continuous distribution for random numbers. A continuous random distribution on the range [min, max) with equal probability throughout the range. The URNG should be real-valued and deliver number in the range [0, 1).

## 5.727 `std::uniform_real_distribution<_RealType>` Class Template Reference

Definition at line 1777 of file random.h.

### 5.727.2 Member Typedef Documentation

**5.727.2.1** `template<typename _RealType = double> typedef _RealType  
std::uniform_real_distribution<_RealType>::result_type`

The type of the range of the distribution.

Definition at line 1784 of file random.h.

### 5.727.3 Constructor & Destructor Documentation

**5.727.3.1** `template<typename _RealType = double> std::uniform_real_-  
distribution<_RealType>::uniform_real_distribution ( _RealType  
__a = _RealType(0), _RealType __b = _RealType(1) )  
[inline, explicit]`

Constructs a `uniform_real_distribution` object.

#### Parameters

`__min` [IN] The lower bound of the distribution.  
`__max` [IN] The upper bound of the distribution.

Definition at line 1823 of file random.h.

### 5.727.4 Member Function Documentation

**5.727.4.1** `template<typename _RealType = double> result_type  
std::uniform_real_distribution<_RealType>::max ( ) const  
[inline]`

Returns the inclusive upper bound of the distribution range.

Definition at line 1875 of file random.h.

**5.727.4.2** `template<typename _RealType = double> result_type  
std::uniform_real_distribution<_RealType>::min ( ) const  
[inline]`

## 5.727 `std::uniform_real_distribution<_RealType>` Class Template Reference 3349

Returns the inclusive lower bound of the distribution range.

Definition at line 1868 of file `random.h`.

**5.727.4.3** `template<typename _RealType = double> template<typename  
_UniformRandomNumberGenerator > result_type  
std::uniform_real_distribution<_RealType>::operator() (  
_UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 1883 of file `random.h`.

References `std::uniform_real_distribution<_RealType>::operator()()`, and  
`std::uniform_real_distribution<_RealType>::param()`.

Referenced by `std::uniform_real_distribution<_RealType>::operator()()`.

**5.727.4.4** `template<typename _RealType = double> void  
std::uniform_real_distribution<_RealType>::param ( const  
param_type & __param ) [inline]`

Sets the parameter set of the distribution.

### Parameters

`__param` The new parameter set of the distribution.

Definition at line 1861 of file `random.h`.

**5.727.4.5** `template<typename _RealType = double> param_type  
std::uniform_real_distribution<_RealType>::param ( ) const  
[inline]`

Returns the parameter set of the distribution.

Definition at line 1853 of file `random.h`.

Referenced by `std::uniform_real_distribution<_RealType>::operator()()`,  
`std::operator==( )`, and `std::operator>>()`.



5.727.4.6 `template<typename _RealType = double> void  
std::uniform_real_distribution<_RealType>::reset ( )  
[inline]`

Resets the distribution state.

Does nothing for the uniform real distribution.

Definition at line 1839 of file `random.h`.

The documentation for this class was generated from the following file:

- [random.h](#)

## 5.728 `std::uniform_real_distribution<_RealType>::param_type` Struct Reference

### Public Types

- typedef `uniform_real_distribution<_RealType>` `distribution_type`

### Public Member Functions

- `param_type` (`_RealType __a=_RealType(0), _RealType __b=_RealType(1)`)
- `result_type a` () const
- `result_type b` () const

### Friends

- bool `operator==` (const `param_type` &\_\_p1, const `param_type` &\_\_p2)

### 5.728.1 Detailed Description

`template<typename _RealType = double> struct std::uniform_real_distribution<_RealType>::param_type`

Parameter type.

Definition at line 1786 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

**5.729 `std::unique_lock<_Mutex>` Class Template Reference**[unique\\_lock](#)**Public Types**

- typedef `_Mutex` **mutex\_type**

**Public Member Functions**

- **unique\_lock** (mutex\_type &\_\_m)
- **unique\_lock** (mutex\_type &\_\_m, [try\\_to\\_lock\\_t](#))
- **unique\_lock** (const [unique\\_lock](#) &)
- **unique\_lock** (mutex\_type &\_\_m, [adopt\\_lock\\_t](#))
- **unique\_lock** ([unique\\_lock](#) &&\_\_u)
- **unique\_lock** (mutex\_type &\_\_m, [defer\\_lock\\_t](#))
- template<typename \_Clock, typename \_Duration >  
  **unique\_lock** (mutex\_type &\_\_m, const [chrono::time\\_point](#)<\_Clock, \_Duration > &\_\_atime)
- template<typename \_Rep, typename \_Period >  
  **unique\_lock** (mutex\_type &\_\_m, const [chrono::duration](#)<\_Rep, \_Period > &\_\_rtime)
- void **lock** ()
- mutex\_type \* **mutex** () const
- **operator bool** () const
- [unique\\_lock](#) & **operator=** (const [unique\\_lock](#) &)
- [unique\\_lock](#) & **operator=** ([unique\\_lock](#) &&\_\_u)
- bool **owns\_lock** () const
- mutex\_type \* **release** ()
- void **swap** ([unique\\_lock](#) &\_\_u)
- bool **try\_lock** ()
- template<typename \_Rep, typename \_Period >  
  bool **try\_lock\_for** (const [chrono::duration](#)<\_Rep, \_Period > &\_\_rtime)
- template<typename \_Clock, typename \_Duration >  
  bool **try\_lock\_until** (const [chrono::time\\_point](#)<\_Clock, \_Duration > &\_\_atime)
- void **unlock** ()

### 5.729.1 Detailed Description

```
template<typename _Mutex> class std::unique_lock<_Mutex >
```

[unique\\_lock](#)

Definition at line 462 of file `mutex`.

The documentation for this class was generated from the following file:

- [mutex](#)

## 5.730 `std::unique_ptr<_Tp, _Dp>` Class Template Reference

20.7.12.2 [unique\\_ptr](#) for single objects.

### Public Types

- typedef `_Dp deleter_type`
- typedef `_Tp element_type`
- typedef `_Pointer::type pointer`

### Public Member Functions

- `unique_ptr` (pointer `__p`)
- `unique_ptr` (pointer `__p`, typename `std::remove_reference< deleter_type >::type &&__d`)
- constexpr `unique_ptr` (nullptr\_t)
- `unique_ptr` (const [unique\\_ptr](#) &)
- `unique_ptr` (pointer `__p`, typename `std::conditional< std::is_reference< deleter_type >::value, deleter_type, const deleter_type & >::type __d`)
- `unique_ptr` ([unique\\_ptr](#) &&\_\_u)
- template<typename `_Up` , typename `_Ep` , typename = typename `std::enable_if< std::is_convertible<typename unique_ptr<_Up, _Ep>::pointer, pointer>::value && !std::is_array<_Up>::value && ((std::is_reference<_Dp>::value && std::is_same<_Ep, _Dp>::value) || (!std::is_reference<_Dp>::value && std::is_convertible<_Ep, _Dp>::value))>::type>  
unique_ptr (unique\_ptr< _Up, _Ep > &&__u)`
- pointer `get` () const
- const deleter\_type & `get_deleter` () const
- deleter\_type & `get_deleter` ()
- `operator bool` () const
- `std::add_lvalue_reference< element_type >::type operator*` () const
- pointer `operator->` () const

- `unique_ptr` & **operator=** (const `unique_ptr` &)
- `unique_ptr` & **operator=** (`unique_ptr` &&\_\_u)
- `unique_ptr` & **operator=** (nullptr\_t)
- template<typename \_Up , typename \_Ep , typename = typename std::enable\_if <std::is\_convertible<typename unique\_ptr<\_Up, \_Ep>::pointer, pointer>::value && !std::is\_array<\_Up>::value>::type>  
`unique_ptr` & **operator=** (`unique_ptr`<\_Up, \_Ep> &&\_\_u)
- pointer **release** ()
- void **reset** (pointer \_\_p=pointer())
- void **swap** (`unique_ptr` &\_\_u)

### 5.730.1 Detailed Description

template<typename \_Tp, typename \_Dp = default\_delete<\_Tp>> class  
`std::unique_ptr<_Tp,_Dp>`

20.7.12.2 `unique_ptr` for single objects.

Definition at line 86 of file `unique_ptr.h`.

The documentation for this class was generated from the following file:

- [unique\\_ptr.h](#)

## 5.731 `std::unique_ptr<_Tp[],_Dp>` Class Template Reference

20.7.12.3 `unique_ptr` for array objects with a runtime length

### Public Types

- typedef \_Dp **deleter\_type**
- typedef \_Tp **element\_type**
- typedef \_Tp \* **pointer**

### Public Member Functions

- `unique_ptr` (pointer \_\_p)
- template<typename \_Up >  
`unique_ptr` (\_Up \*, typename [std::remove\\_reference](#)< `deleter_type` >::type  
&&, typename [std::enable\\_if](#)< [std::is\\_convertible](#)< \_Up \*, pointer >::value  
>::type !=0)
- `unique_ptr` (pointer \_\_p, typename [std::remove\\_reference](#)< `deleter_type`  
>::type &&\_\_d)

- constexpr `unique_ptr` (`nullptr_t`)
- `unique_ptr` (`const unique_ptr &`)
- `unique_ptr` (`pointer __p`, `typename std::conditional< std::is_reference< deleter_type >::value, deleter_type, const deleter_type & >::type __d`)
- `unique_ptr` (`unique_ptr &&__u`)
- `template<typename _Up >`  
`unique_ptr` (`_Up *`, `typename std::conditional< std::is_reference< deleter_type >::value, deleter_type, const deleter_type & >::type, typename std::enable_if< std::is_convertible< _Up *, pointer >::value >::type !=0`)
- `template<typename _Up >`  
`unique_ptr` (`_Up *`, `typename std::enable_if< std::is_convertible< _Up *, pointer >::value >::type !=0`)
- `template<typename _Up, typename _Ep >`  
`unique_ptr` (`unique_ptr< _Up, _Ep > &&__u`)
- `pointer get () const`
- `deleter_type & get_deleter ()`
- `const deleter_type & get_deleter () const`
- `operator bool () const`
- `unique_ptr & operator= (nullptr_t)`
- `template<typename _Up, typename _Ep >`  
`unique_ptr & operator= (unique_ptr< _Up, _Ep > &&__u)`
- `unique_ptr & operator= (unique_ptr &&__u)`
- `unique_ptr & operator= (const unique_ptr &)`
- `std::add_lvalue_reference< element_type >::type operator[] (size_t __i) const`
- `pointer release ()`
- `void reset (pointer __p=pointer())`
- `void reset (nullptr_t)`
- `template<typename _Up >`  
`void reset (_Up)`
- `void swap (unique_ptr &&__u)`

### 5.731.1 Detailed Description

`template<typename _Tp, typename _Dp> class std::unique_ptr<_Tp[],_Dp>`

20.7.12.3 `unique_ptr` for array objects with a runtime length

Definition at line 263 of file `unique_ptr.h`.

The documentation for this class was generated from the following file:

- `unique_ptr.h`

### 5.732 `std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >` Class Template Reference

A standard container composed of unique keys (containing at most one of each key value) that associates values of another type with the keys.

Inherits `std::__unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`.

#### Public Types

- `typedef _Base::allocator_type allocator_type`
- `typedef __detail::__Hashtable_const_iterator< value_type, __constant_iterators, __cache_hash_code > const_iterator`
- `typedef __detail::_Node_const_iterator< value_type, __constant_iterators, __cache_hash_code > const_local_iterator`
- `typedef _Allocator::const_pointer const_pointer`
- `typedef _Allocator::const_reference const_reference`
- `typedef std::ptrdiff_t difference_type`
- `typedef _Base::hasher hasher`
- `typedef __detail::__Hashtable_iterator< value_type, __constant_iterators, __cache_hash_code > iterator`
- `typedef _Base::key_equal key_equal`
- `typedef _Key key_type`
- `typedef __detail::_Node_iterator< value_type, __constant_iterators, __cache_hash_code > local_iterator`
- `typedef _Allocator::pointer pointer`
- `typedef _Allocator::reference reference`
- `typedef _Base::size_type size_type`
- `typedef \_Base::value\_type value_type`

#### Public Member Functions

- **unordered\_map** (size\_type \_\_n=10, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- `template<typename _InputIterator >`  
**unordered\_map** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_map** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- `const _RehashPolicy & __rehash_policy () const`
- `void __rehash_policy (const _RehashPolicy &)`

- `local_iterator` **begin** (`size_type __n`)
- `const_local_iterator` **begin** (`size_type __n`) `const`
- `iterator` **begin** ()
- `const_iterator` **begin** () `const`
- `size_type` **bucket** (`const key_type &__k`) `const`
- `size_type` **bucket\_count** () `const`
- `size_type` **bucket\_size** (`size_type __n`) `const`
- `const_iterator` **cbegin** () `const`
- `const_local_iterator` **cbegin** (`size_type __n`) `const`
- `const_iterator` **cend** () `const`
- `const_local_iterator` **cend** (`size_type`) `const`
- `void` **clear** ()
- `size_type` **count** (`const key_type &__k`) `const`
- `bool` **empty** () `const`
- `iterator` **end** ()
- `local_iterator` **end** (`size_type`)
- `const_local_iterator` **end** (`size_type`) `const`
- `const_iterator` **end** () `const`
- `std::pair`< `iterator`, `iterator` > **equal\_range** (`const key_type &__k`)
- `std::pair`< `const_iterator`, `const_iterator` > **equal\_range** (`const key_type &__k`) `const`
- `iterator` **erase** (`const_iterator`)
- `size_type` **erase** (`const key_type &`)
- `iterator` **erase** (`const_iterator`, `const_iterator`)
- `const_iterator` **find** (`const key_type &__k`) `const`
- `iterator` **find** (`const key_type &__k`)
- `allocator_type` **get\_allocator** () `const`
- `void` **insert** (`initializer_list`< `value_type` > `__l`)
- `template`< `typename` `_InputIterator` >  
  `void` **insert** (`_InputIterator __first`, `_InputIterator __last`)
- `iterator` **insert** (`const_iterator`, `const value_type &__v`)
- `template`< `typename` `_Pair`, `typename` `= typename std::enable_if<!__constant_iterators && std::is_convertible<_Pair, value_type>::value>::type` >  
  `iterator` **insert** (`const_iterator`, `_Pair &&__v`)
- `_Insert_Return_Type` **insert** (`const value_type &__v`)
- `iterator` **insert** (`const_iterator`, `value_type &&__v`)
- `template`< `typename` `_Pair`, `typename` `= typename std::enable_if<!__constant_iterators && std::is_convertible<_Pair, value_type>::value>::type` >  
  `_Insert_Return_Type` **insert** (`_Pair &&__v`)
- `_Insert_Return_Type` **insert** (`value_type &&__v`)
- `key_equal` **key\_eq** () `const`
- `float` **load\_factor** () `const`
- `size_type` **max\_bucket\_count** () `const`

- `size_type max_size ()` const
- `unordered_map` & `operator=` (`initializer_list`< `value_type` > \_\_l)
- `void rehash (size_type __n)`
- `size_type size ()` const
- `void swap (_Hashtable &)`

#### Friends

- `struct __detail::_Map_base`

#### 5.732.1 Detailed Description

```
template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred =
std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>
>> class std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >
```

A standard container composed of unique keys (containing at most one of each key value) that associates values of another type with the keys. Meets the requirements of a [container](#), and [unordered associative container](#)

#### Parameters

- Key* Type of key objects.
- Tp* Type of mapped objects.
- Hash* Hashing function object type, defaults to `hash<Value>`.
- Pred* Predicate function object type, defaults to `equal_to<Value>`.
- Alloc* Allocator type, defaults to `allocator<Key>`.

The resulting value type of the container is `std::pair<const Key, Tp>`.

Definition at line 256 of file `unordered_map.h`.

The documentation for this class was generated from the following file:

- [unordered\\_map.h](#)

#### 5.733 `std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>` Class Template Reference

A standard container composed of equivalent keys (possibly containing multiple of each key value) that associates values of another type with the keys.

Inherits `std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>`.



## Public Types

- `typedef _Base::allocator_type allocator_type`
- `typedef __detail::_Hashtable_const_iterator< value_type, __constant_iterators, __cache_hash_code > const_iterator`
- `typedef __detail::_Node_const_iterator< value_type, __constant_iterators, __cache_hash_code > const_local_iterator`
- `typedef _Allocator::const_pointer const_pointer`
- `typedef _Allocator::const_reference const_reference`
- `typedef std::ptrdiff_t difference_type`
- `typedef _Base::hasher hasher`
- `typedef __detail::_Hashtable_iterator< value_type, __constant_iterators, __cache_hash_code > iterator`
- `typedef _Base::key_equal key_equal`
- `typedef _Key key_type`
- `typedef __detail::_Node_iterator< value_type, __constant_iterators, __cache_hash_code > local_iterator`
- `typedef _Allocator::pointer pointer`
- `typedef _Allocator::reference reference`
- `typedef _Base::size_type size_type`
- `typedef \_Base::value\_type value_type`

## Public Member Functions

- **unordered\_multimap** (size\_type \_\_n=10, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- `template<typename _InputIterator >`  
**unordered\_multimap** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_multimap** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- const `_RehashPolicy` & **\_\_rehash\_policy** () const
- void **\_\_rehash\_policy** (const `_RehashPolicy` &)
- local\_iterator **begin** (size\_type \_\_n)
- const\_local\_iterator **begin** (size\_type \_\_n) const
- iterator **begin** ()
- const\_iterator **begin** () const
- size\_type **bucket** (const key\_type &\_\_k) const
- size\_type **bucket\_count** () const
- size\_type **bucket\_size** (size\_type \_\_n) const
- const\_iterator **cbegin** () const

- `const_local_iterator` **cbegin** (size\_type \_\_n) const
- `const_iterator` **end** () const
- `const_local_iterator` **end** (size\_type) const
- void **clear** ()
- size\_type **count** (const key\_type &\_\_k) const
- bool **empty** () const
- iterator **end** ()
- local\_iterator **end** (size\_type)
- `const_local_iterator` **end** (size\_type) const
- `const_iterator` **end** () const
- [std::pair](#)< iterator, iterator > **equal\_range** (const key\_type &\_\_k)
- [std::pair](#)< const\_iterator, const\_iterator > **equal\_range** (const key\_type &\_\_k) const
- iterator **erase** (const\_iterator)
- size\_type **erase** (const key\_type &)
- iterator **erase** (const\_iterator, const\_iterator)
- `const_iterator` **find** (const key\_type &\_\_k) const
- iterator **find** (const key\_type &\_\_k)
- allocator\_type **get\_allocator** () const
- void **insert** ([initializer\\_list](#)< value\_type > \_\_l)
- template<typename \_InputIterator >  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- iterator **insert** (const\_iterator, const value\_type &\_\_v)
- template<typename \_Pair, typename = typename std::enable\_if<!\_\_constant\_iterators && std::is\_convertible<\_Pair, value\_type>::value>::type>  
iterator **insert** (const\_iterator, \_Pair &&\_\_v)
- [\\_Insert\\_Return\\_Type](#) **insert** (const value\_type &\_\_v)
- iterator **insert** (const\_iterator, value\_type &&\_\_v)
- template<typename \_Pair, typename = typename std::enable\_if<!\_\_constant\_iterators && std::is\_convertible<\_Pair, value\_type>::value>::type>  
[\\_Insert\\_Return\\_Type](#) **insert** (\_Pair &&\_\_v)
- [\\_Insert\\_Return\\_Type](#) **insert** (value\_type &&\_\_v)
- key\_equal **key\_eq** () const
- float **load\_factor** () const
- size\_type **max\_bucket\_count** () const
- size\_type **max\_size** () const
- `unordered_multimap` & **operator=** ([initializer\\_list](#)< value\_type > \_\_l)
- void **rehash** (size\_type \_\_n)
- size\_type **size** () const
- void **swap** (\_Hashtable &)

## Friends

- struct `__detail::_Map_base`

### 5.733.1 Detailed Description

```
template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred =
std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>
>> class std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >
```

A standard container composed of equivalent keys (possibly containing multiple of each key value) that associates values of another type with the keys. Meets the requirements of a [container](#), and [unordered associative container](#)

#### Parameters

*Key* Type of key objects.

*Tp* Type of mapped objects.

*Hash* Hashing function object type, defaults to `hash<Value>`.

*Pred* Predicate function object type, defaults to `equal_to<Value>`.

*Alloc* Allocator type, defaults to `allocator<Key>`.

The resulting value type of the container is `std::pair<const Key, Tp>`.

Definition at line 324 of file `unordered_map.h`.

The documentation for this class was generated from the following file:

- [unordered\\_map.h](#)

### 5.734 `std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >` Class Template Reference

A standard container composed of equivalent keys (possibly containing multiple of each key value) in which the elements' keys are the elements themselves.

Inherits `std::__unordered_multiset< _Value, _Hash, _Pred, _Alloc >`.

#### Public Types

- `typedef _Base::allocator_type allocator_type`
- `typedef __detail::__Hashtable_const_iterator< value_type, __constant_iterators, __cache_hash_code > const_iterator`
- `typedef __detail::__Node_const_iterator< value_type, __constant_iterators, __cache_hash_code > const_local_iterator`
- `typedef _Allocator::const_pointer const_pointer`
- `typedef _Allocator::const_reference const_reference`
- `typedef std::ptrdiff_t difference_type`

- `typedef _Base::hasher` **hasher**
- `typedef __detail::_Hashtable_iterator< value_type, __constant_iterators, __cache_hash_code >` **iterator**
- `typedef _Base::key_equal` **key\_equal**
- `typedef _Key` **key\_type**
- `typedef __detail::_Node_iterator< value_type, __constant_iterators, __cache_hash_code >` **local\_iterator**
- `typedef _Allocator::pointer` **pointer**
- `typedef _Allocator::reference` **reference**
- `typedef _Base::size_type` **size\_type**
- `typedef _Base::value_type` **value\_type**

## Public Member Functions

- **unordered\_multiset** (size\_type \_\_n=10, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- `template<typename _InputIterator >`  
**unordered\_multiset** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_multiset** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- `const _RehashPolicy & __rehash_policy ()` const
- `void __rehash_policy (const _RehashPolicy &)`
- `local_iterator` **begin** (size\_type \_\_n)
- `const_local_iterator` **begin** (size\_type \_\_n) const
- `iterator` **begin** ()
- `const_iterator` **begin** () const
- `size_type` **bucket** (const key\_type &\_\_k) const
- `size_type` **bucket\_count** () const
- `size_type` **bucket\_size** (size\_type \_\_n) const
- `const_iterator` **cbegin** () const
- `const_local_iterator` **cbegin** (size\_type \_\_n) const
- `const_iterator` **cend** () const
- `const_local_iterator` **cend** (size\_type) const
- `void` **clear** ()
- `size_type` **count** (const key\_type &\_\_k) const
- `bool` **empty** () const
- `iterator` **end** ()
- `local_iterator` **end** (size\_type)
- `const_local_iterator` **end** (size\_type) const
- `const_iterator` **end** () const

- `std::pair< iterator, iterator > equal_range` (const key\_type &\_\_k)
- `std::pair< const_iterator, const_iterator > equal_range` (const key\_type &\_\_k) const
- iterator `erase` (const\_iterator)
- size\_type `erase` (const key\_type &)
- iterator `erase` (const\_iterator, const\_iterator)
- const\_iterator `find` (const key\_type &\_\_k) const
- iterator `find` (const key\_type &\_\_k)
- allocator\_type `get_allocator` () const
- void `insert` (initializer\_list< value\_type > \_\_l)
- template<typename \_InputIterator >  
void `insert` (\_InputIterator \_\_first, \_InputIterator \_\_last)
- iterator `insert` (const\_iterator, const value\_type &\_\_v)
- template<typename \_Pair, typename = typename std::enable\_if<!\_\_constant\_iterators && std::is\_convertible<\_Pair, value\_type>::value>::type>  
iterator `insert` (const\_iterator, \_Pair &&\_\_v)
- `_Insert_Return_Type insert` (const value\_type &\_\_v)
- iterator `insert` (const\_iterator, value\_type &&\_\_v)
- template<typename \_Pair, typename = typename std::enable\_if<!\_\_constant\_iterators && std::is\_convertible<\_Pair, value\_type>::value>::type>  
`_Insert_Return_Type insert` (\_Pair &&\_\_v)
- `_Insert_Return_Type insert` (value\_type &&\_\_v)
- key\_equal `key_eq` () const
- float `load_factor` () const
- size\_type `max_bucket_count` () const
- size\_type `max_size` () const
- `unordered_multiset & operator=` (initializer\_list< value\_type > \_\_l)
- void `rehash` (size\_type \_\_n)
- size\_type `size` () const
- void `swap` (\_Hashtable &)

## Friends

- struct `__detail::_Map_base`

### 5.734.1 Detailed Description

`template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> class std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>`

A standard container composed of equivalent keys (possibly containing multiple of each key value) in which the elements' keys are the elements themselves. Meets the requirements of a [container](#), and [unordered associative container](#)

### Parameters

- Value** Type of key objects.
- Hash** Hashing function object type, defaults to `hash<Value>`.
- Pred** Predicate function object type, defaults to `equal_to<Value>`.
- Alloc** Allocator type, defaults to `allocator<Key>`.

Definition at line 314 of file `unordered_set.h`.

The documentation for this class was generated from the following file:

- [unordered\\_set.h](#)

### 5.735 `std::unordered_set< _Value, _Hash, _Pred, _Alloc >` Class Template Reference

A standard container composed of unique keys (containing at most one of each key value) in which the elements' keys are the elements themselves.

Inherits `std::__unordered_set< _Value, _Hash, _Pred, _Alloc >`.

### Public Types

- `typedef _Base::allocator_type allocator_type`
- `typedef __detail::_Hashtable_const_iterator< value_type, __constant_iterators, __cache_hash_code > const_iterator`
- `typedef __detail::_Node_const_iterator< value_type, __constant_iterators, __cache_hash_code > const_local_iterator`
- `typedef _Allocator::const_pointer const_pointer`
- `typedef _Allocator::const_reference const_reference`
- `typedef std::ptrdiff_t difference_type`
- `typedef _Base::hasher hasher`
- `typedef __detail::_Hashtable_iterator< value_type, __constant_iterators, __cache_hash_code > iterator`
- `typedef _Base::key_equal key_equal`
- `typedef _Key key_type`
- `typedef __detail::_Node_iterator< value_type, __constant_iterators, __cache_hash_code > local_iterator`
- `typedef _Allocator::pointer pointer`
- `typedef _Allocator::reference reference`
- `typedef _Base::size_type size_type`
- `typedef _Base::value_type value_type`

## Public Member Functions

- **unordered\_set** (size\_type \_\_n=10, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator >  
**unordered\_set** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_set** (initializer\_list< value\_type > \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- const \_RehashPolicy & **\_\_rehash\_policy** () const
- void **\_\_rehash\_policy** (const \_RehashPolicy &)
- local\_iterator **begin** (size\_type \_\_n)
- const\_local\_iterator **begin** (size\_type \_\_n) const
- iterator **begin** ()
- const\_iterator **begin** () const
- size\_type **bucket** (const key\_type &\_\_k) const
- size\_type **bucket\_count** () const
- size\_type **bucket\_size** (size\_type \_\_n) const
- const\_iterator **cbegin** () const
- const\_local\_iterator **cbegin** (size\_type \_\_n) const
- const\_iterator **cend** () const
- const\_local\_iterator **cend** (size\_type) const
- void **clear** ()
- size\_type **count** (const key\_type &\_\_k) const
- bool **empty** () const
- iterator **end** ()
- local\_iterator **end** (size\_type)
- const\_local\_iterator **end** (size\_type) const
- const\_iterator **end** () const
- [std::pair](#)< iterator, iterator > **equal\_range** (const key\_type &\_\_k)
- [std::pair](#)< const\_iterator, const\_iterator > **equal\_range** (const key\_type &\_\_k) const
- iterator **erase** (const\_iterator)
- size\_type **erase** (const key\_type &)
- iterator **erase** (const\_iterator, const\_iterator)
- const\_iterator **find** (const key\_type &\_\_k) const
- iterator **find** (const key\_type &\_\_k)
- allocator\_type **get\_allocator** () const
- void **insert** (initializer\_list< value\_type > \_\_l)
- template<typename \_InputIterator >  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)

- iterator **insert** (const\_iterator, const value\_type &\_\_v)
- template<typename \_Pair, typename = typename std::enable\_if<!\_\_constant\_iterators && std::is\_convertible<\_Pair, value\_type>::value>::type>  
iterator **insert** (const\_iterator, \_Pair &&\_\_v)
- [\\_Insert\\_Return\\_Type insert](#) (const value\_type &\_\_v)
- iterator **insert** (const\_iterator, value\_type &&\_\_v)
- template<typename \_Pair, typename = typename std::enable\_if<!\_\_constant\_iterators && std::is\_convertible<\_Pair, value\_type>::value>::type>  
[\\_Insert\\_Return\\_Type insert](#) (\_Pair &&\_\_v)
- [\\_Insert\\_Return\\_Type insert](#) (value\_type &&\_\_v)
- key\_equal **key\_eq** () const
- float **load\_factor** () const
- size\_type **max\_bucket\_count** () const
- size\_type **max\_size** () const
- [unordered\\_set](#) & **operator=** ([initializer\\_list](#)< value\_type > \_\_l)
- void **rehash** (size\_type \_\_n)
- size\_type **size** () const
- void **swap** (\_Hashtable &)

## Friends

- struct `__detail::_Map_base`

### 5.735.1 Detailed Description

`template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> class std::unordered_set<_Value, _Hash, _Pred, _Alloc>`

A standard container composed of unique keys (containing at most one of each key value) in which the elements' keys are the elements themselves. Meets the requirements of a [container](#), and [unordered associative container](#)

## Parameters

- Value** Type of key objects.
- Hash** Hashing function object type, defaults to `hash<Value>`.
- Pred** Predicate function object type, defaults to `equal_to<Value>`.
- Alloc** Allocator type, defaults to `allocator<Key>`.

Definition at line 249 of file `unordered_set.h`.

The documentation for this class was generated from the following file:

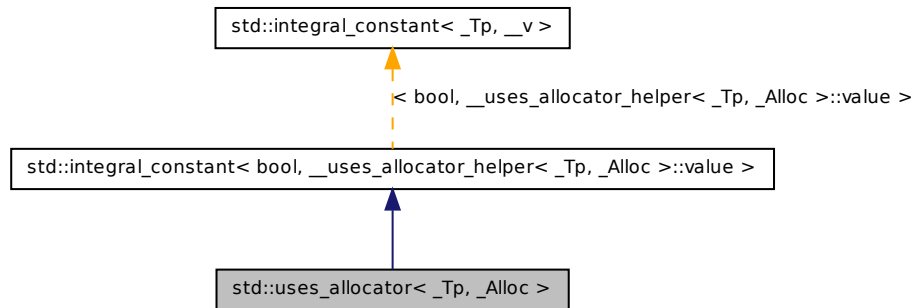
- [unordered\\_set.h](#)



## 5.736 `std::uses_allocator<_Tp, _Alloc>` Struct Template Reference

[allocator.uses.trait]

Inheritance diagram for `std::uses_allocator<_Tp, _Alloc>`:



### Public Types

- typedef `integral_constant<bool, __v>` **type**
- typedef `bool` **value\_type**

### Public Member Functions

- constexpr **operator value\_type** ()

### Static Public Attributes

- static constexpr `bool` **value**

#### 5.736.1 Detailed Description

**template<typename \_Tp, typename \_Alloc> struct `std::uses_allocator<_Tp, _Alloc>`**

[allocator.uses.trait]

Definition at line 230 of file `allocator.h`.

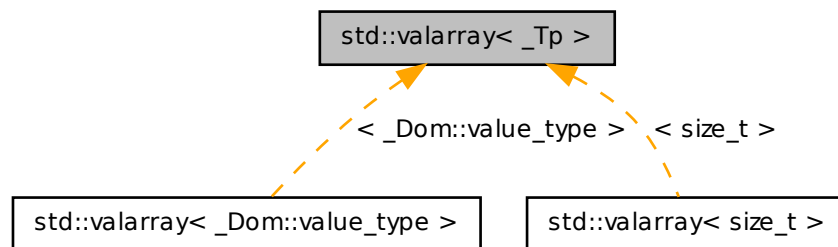
The documentation for this struct was generated from the following file:

- [allocator.h](#)

### 5.737 `std::valarray<_Tp>` Class Template Reference

Smart array designed to support numeric processing.

Inheritance diagram for `std::valarray<_Tp>`:



#### Public Types

- `typedef _Tp value_type`

#### Public Member Functions

- [valarray](#) ()
- [valarray](#) (size\_t)
- [valarray](#) (const \_Tp \*\_\_restrict\_\_, size\_t)
- [valarray](#) (const [mask\\_array](#)<\_Tp> &)
- [valarray](#) (const [indirect\\_array](#)<\_Tp> &)
- `template<typename _Tp>`  
`valarray` (const \_Tp \*\_\_restrict\_\_ \_\_p, size\_t \_\_n)
- [valarray](#) (const [valarray](#) &)
- [valarray](#) ([initializer\\_list](#)<\_Tp>)

- template<class \_Dom >  
**valarray** (const \_Expr< \_Dom, \_Tp > &\_\_e)
- **valarray** (const \_Tp &, size\_t)
- **valarray** (const [slice\\_array](#)< \_Tp > &)
- **valarray** (const [gslice\\_array](#)< \_Tp > &)
- \_Expr< \_ValFunClos< \_ValArray, \_Tp >, \_Tp > [apply](#) (\_Tp func(\_Tp)) const
- \_Expr< \_RefFunClos< \_ValArray, \_Tp >, \_Tp > [apply](#) (\_Tp func(const \_Tp &)) const
- **valarray**< \_Tp > [cshift](#) (int) const
- \_Tp [max](#) () const
- \_Tp [min](#) () const
- \_UnaryOp< \_\_logical\_not >::\_Rt [operator!](#) () const
- **valarray**< \_Tp > & [operator%=>](#) (const \_Tp &)
- **valarray**< \_Tp > & [operator%=>](#) (const **valarray**< \_Tp > &)
- template<class \_Dom >  
**valarray**< \_Tp > & [operator%=>](#) (const \_Expr< \_Dom, \_Tp > &)
- **valarray**< \_Tp > & [operator&=>](#) (const \_Tp &)
- **valarray**< \_Tp > & [operator&=>](#) (const **valarray**< \_Tp > &)
- template<class \_Dom >  
**valarray**< \_Tp > & [operator&=>](#) (const \_Expr< \_Dom, \_Tp > &)
- **valarray**< \_Tp > & [operator\\*=>](#) (const \_Tp &)
- **valarray**< \_Tp > & [operator\\*=>](#) (const **valarray**< \_Tp > &)
- template<class \_Dom >  
**valarray**< \_Tp > & [operator\\*=>](#) (const \_Expr< \_Dom, \_Tp > &)
- \_UnaryOp< \_\_unary\_plus >::\_Rt [operator+>](#) () const
- **valarray**< \_Tp > & [operator+=>](#) (const \_Tp &)
- **valarray**< \_Tp > & [operator+=>](#) (const **valarray**< \_Tp > &)
- template<class \_Dom >  
**valarray**< \_Tp > & [operator+=>](#) (const \_Expr< \_Dom, \_Tp > &)
- \_UnaryOp< \_\_negate >::\_Rt [operator->](#) () const
- **valarray**< \_Tp > & [operator->](#) (const \_Tp &)
- **valarray**< \_Tp > & [operator->](#) (const **valarray**< \_Tp > &)
- template<class \_Dom >  
**valarray**< \_Tp > & [operator->](#) (const \_Expr< \_Dom, \_Tp > &)
- **valarray**< \_Tp > & [operator/=>](#) (const \_Tp &)
- **valarray**< \_Tp > & [operator/=>](#) (const **valarray**< \_Tp > &)
- template<class \_Dom >  
**valarray**< \_Tp > & [operator/=>](#) (const \_Expr< \_Dom, \_Tp > &)
- **valarray**< \_Tp > & [operator<=>](#) (const \_Tp &)
- **valarray**< \_Tp > & [operator<=>](#) (const **valarray**< \_Tp > &)
- template<class \_Dom >  
**valarray**< \_Tp > & [operator<=>](#) (const \_Expr< \_Dom, \_Tp > &)
- **valarray**< \_Tp > & [operator=>](#) (const \_Tp &)

- `valarray<_Tp> & operator= (const gslice_array<_Tp> &)`
- `valarray<_Tp> & operator= (const slice_array<_Tp> &)`
- `valarray<_Tp> & operator= (const indirect_array<_Tp> &)`
- `valarray & operator= (initializer_list<_Tp>)`
- `template<class _Dom>`  
`valarray<_Tp> & operator= (const _Expr<_Dom, _Tp> &)`
- `valarray<_Tp> & operator= (const mask_array<_Tp> &)`
- `valarray<_Tp> & operator= (const valarray<_Tp> &)`
- `valarray<_Tp> & operator>>= (const valarray<_Tp> &)`
- `template<class _Dom>`  
`valarray<_Tp> & operator>>= (const _Expr<_Dom, _Tp> &)`
- `valarray<_Tp> & operator>>= (const _Tp &)`
- `gslice_array<_Tp> operator[] (const gslice &)`
- `_Tp & operator[] (size_t)`
- `_Expr<_SClos<_ValArray, _Tp>, _Tp> operator[] (slice) const`
- `valarray<_Tp> operator[] (const valarray<bool> &) const`
- `_Expr<_IClos<_ValArray, _Tp>, _Tp> operator[] (const valarray<size_t> &) const`
- `slice_array<_Tp> operator[] (slice)`
- `_Expr<_GClos<_ValArray, _Tp>, _Tp> operator[] (const gslice &) const`
- `indirect_array<_Tp> operator[] (const valarray<size_t> &)`
- `const _Tp & operator[] (size_t) const`
- `mask_array<_Tp> operator[] (const valarray<bool> &)`
- `valarray<_Tp> & operator^= (const valarray<_Tp> &)`
- `template<class _Dom>`  
`valarray<_Tp> & operator^= (const _Expr<_Dom, _Tp> &)`
- `valarray<_Tp> & operator^= (const _Tp &)`
- `template<class _Dom>`  
`valarray<_Tp> & operator|= (const _Expr<_Dom, _Tp> &)`
- `valarray<_Tp> & operator|= (const _Tp &)`
- `valarray<_Tp> & operator|= (const valarray<_Tp> &)`
- `_UnaryOp<__bitwise_not>::_Rt operator~ () const`
- `void resize (size_t __size, _Tp __c=_Tp())`
- `valarray<_Tp> shift (int) const`
- `size_t size () const`
- `_Tp sum () const`

## Friends

- `class _Array<_Tp>`

### 5.737.1 Detailed Description

**template<class \_Tp> class std::valarray< \_Tp >**

Smart array designed to support numeric processing. A valarray is an array that provides constraints intended to allow for effective optimization of numeric array processing by reducing the aliasing that can result from pointer representations. It represents a one-dimensional array from which different multidimensional subsets can be accessed and modified.

#### Parameters

*Tp* Type of object in the array.

Definition at line 116 of file valarray.

### 5.737.2 Constructor & Destructor Documentation

**5.737.2.1** **template<class \_Tp> std::valarray< \_Tp >::valarray ( const \_Tp \* \_\_restrict\_\_, size\_t )**

Construct an array initialized to the first  $n$  elements of  $t$ .

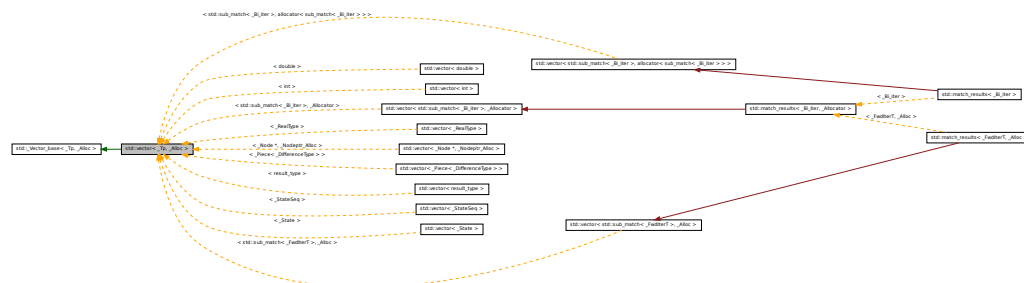
The documentation for this class was generated from the following file:

- [valarray](#)

## 5.738 std::vector< \_Tp, \_Alloc > Class Template Reference

A standard container which offers fixed time access to individual elements in any order.

Inheritance diagram for std::vector< \_Tp, \_Alloc >:



**Public Types**

- typedef `_Alloc` **allocator\_type**
- typedef `__gnu_cxx::__normal_iterator< const_pointer, vector>` **const\_iterator**
- typedef `_Tp_alloc_type::const_pointer` **const\_pointer**
- typedef `_Tp_alloc_type::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `ptrdiff_t` **difference\_type**
- typedef `__gnu_cxx::__normal_iterator< pointer, vector>` **iterator**
- typedef `_Tp_alloc_type::pointer` **pointer**
- typedef `_Tp_alloc_type::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `size_t` **size\_type**
- typedef `_Tp` **value\_type**

**Public Member Functions**

- [vector](#) ()
- [vector](#) (const allocator\_type &\_\_a)
- [vector](#) (size\_type \_\_n, const value\_type &\_\_value, const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator >  
[vector](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, const allocator\_type &\_\_a=allocator\_type())
- [vector](#) (const [vector](#) &\_\_x)
- [vector](#) (size\_type \_\_n)
- [vector](#) ([vector](#) &&\_\_x)
- [vector](#) ([initializer\\_list](#)< value\_type > \_\_l, const allocator\_type &\_\_a=allocator\_type())
- [~vector](#) ()
- void [assign](#) (size\_type \_\_n, const value\_type &\_\_val)
- template<typename \_InputIterator >  
void [assign](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void [assign](#) ([initializer\\_list](#)< value\_type > \_\_l)
- reference [at](#) (size\_type \_\_n)
- const\_reference [at](#) (size\_type \_\_n) const
- reference [back](#) ()
- const\_reference [back](#) () const
- iterator [begin](#) ()
- const\_iterator [begin](#) () const
- size\_type [capacity](#) () const
- const\_iterator [cbegin](#) () const

- `const_iterator` `cend` () const
- `void` `clear` ()
- `const_reverse_iterator` `crbegin` () const
- `const_reverse_iterator` `crend` () const
- `_Tp *` `data` ()
- `const _Tp *` `data` () const
- `template<typename... _Args>`  
`iterator` `emplace` (`iterator` \_\_position, `_Args` &&...\_\_args)
- `template<typename... _Args>`  
`void` `emplace_back` (`_Args` &&...\_\_args)
- `bool` `empty` () const
- `iterator` `end` ()
- `const_iterator` `end` () const
- `iterator` `erase` (`iterator` \_\_first, `iterator` \_\_last)
- `iterator` `erase` (`iterator` \_\_position)
- `const_reference` `front` () const
- `reference` `front` ()
- `iterator` `insert` (`iterator` \_\_position, `value_type` &&\_\_x)
- `template<typename _InputIterator>`  
`void` `insert` (`iterator` \_\_position, `_InputIterator` \_\_first, `_InputIterator` \_\_last)
- `void` `insert` (`iterator` \_\_position, `initializer_list`< `value_type` > \_\_l)
- `void` `insert` (`iterator` \_\_position, `size_type` \_\_n, `const value_type` &\_\_x)
- `iterator` `insert` (`iterator` \_\_position, `const value_type` &\_\_x)
- `size_type` `max_size` () const
- `vector` & `operator=` (`initializer_list`< `value_type` > \_\_l)
- `vector` & `operator=` (`vector` &&\_\_x)
- `vector` & `operator=` (`const vector` &\_\_x)
- `reference` `operator[]` (`size_type` \_\_n)
- `const_reference` `operator[]` (`size_type` \_\_n) const
- `void` `pop_back` ()
- `void` `push_back` (`value_type` &&\_\_x)
- `void` `push_back` (`const value_type` &\_\_x)
- `reverse_iterator` `rbegin` ()
- `const_reverse_iterator` `rbegin` () const
- `const_reverse_iterator` `rend` () const
- `reverse_iterator` `rend` ()
- `void` `reserve` (`size_type` \_\_n)
- `void` `resize` (`size_type` \_\_new\_size)
- `void` `resize` (`size_type` \_\_new\_size, `const value_type` &\_\_x)
- `void` `shrink_to_fit` ()
- `size_type` `size` () const
- `void` `swap` (`vector` &\_\_x)

**Protected Member Functions**

- `_Tp_alloc_type::pointer` **\_M\_allocate** (size\_t \_\_n)
- `template<typename _ForwardIterator >`  
`pointer` **\_M\_allocate\_and\_copy** (size\_type \_\_n, \_ForwardIterator \_\_first, \_ForwardIterator \_\_last)
- `template<typename _InputIterator >`  
`void` **\_M\_assign\_aux** (\_InputIterator \_\_first, \_InputIterator \_\_last, `std::input_iterator_tag`)
- `template<typename _ForwardIterator >`  
`void` **\_M\_assign\_aux** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, `std::forward_iterator_tag`)
- `template<typename _Integer >`  
`void` **\_M\_assign\_dispatch** (\_Integer \_\_n, \_Integer \_\_val, \_\_true\_type)
- `template<typename _InputIterator >`  
`void` **\_M\_assign\_dispatch** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- `size_type` **\_M\_check\_len** (size\_type \_\_n, const char \*\_\_s) const
- `void` **\_M\_deallocate** (typename \_Tp\_alloc\_type::pointer \_\_p, size\_t \_\_n)
- `void` **\_M\_default\_append** (size\_type \_\_n)
- `void` **\_M\_default\_initialize** (size\_type \_\_n)
- `void` **\_M\_erase\_at\_end** (pointer \_\_pos)
- `void` **\_M\_fill\_assign** (size\_type \_\_n, const value\_type &\_\_val)
- `void` **\_M\_fill\_initialize** (size\_type \_\_n, const value\_type &\_\_value)
- `void` **\_M\_fill\_insert** (iterator \_\_pos, size\_type \_\_n, const value\_type &\_\_x)
- `_Tp_alloc_type &` **\_M\_get\_Tp\_allocator** ()
- `const _Tp_alloc_type &` **\_M\_get\_Tp\_allocator** () const
- `template<typename _InputIterator >`  
`void` **\_M\_initialize\_dispatch** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- `template<typename _Integer >`  
`void` **\_M\_initialize\_dispatch** (\_Integer \_\_n, \_Integer \_\_value, \_\_true\_type)
- `template<typename... _Args>`  
`void` **\_M\_insert\_aux** (iterator \_\_position, \_Args &&...\_\_args)
- `template<typename _InputIterator >`  
`void` **\_M\_insert\_dispatch** (iterator \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- `template<typename _Integer >`  
`void` **\_M\_insert\_dispatch** (iterator \_\_pos, \_Integer \_\_n, \_Integer \_\_val, \_\_true\_type)
- `void` **\_M\_range\_check** (size\_type \_\_n) const
- `template<typename _ForwardIterator >`  
`void` **\_M\_range\_initialize** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, `std::forward_iterator_tag`)



- `template<typename _InputIterator >`  
`void _M_range_initialize` (`_InputIterator __first`, `_InputIterator __last`,  
`std::input_iterator_tag`)
- `template<typename _ForwardIterator >`  
`void _M_range_insert` (`iterator __pos`, `_ForwardIterator __first`, `_-`  
`ForwardIterator __last`, `std::forward_iterator_tag`)
- `template<typename _InputIterator >`  
`void _M_range_insert` (`iterator __pos`, `_InputIterator __first`, `_InputIterator __-`  
`last`, `std::input_iterator_tag`)
- `allocator_type get_allocator` () const

### Protected Attributes

- `_Vector_impl _M_impl`

#### 5.738.1 Detailed Description

`template<typename _Tp, typename _Alloc = std::allocator<_Tp>> class`  
`std::vector<_Tp, _Alloc>`

A standard container which offers fixed time access to individual elements in any order. Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#), including the [optional sequence requirements](#) with the exception of `push_front` and `pop_front`.

In some terminology a vector can be described as a dynamic C-style array, it offers fast and efficient access to individual elements in any order and saves the user from worrying about memory and size allocation. Subscripting ( `[]` ) access is also provided as with C-style arrays.

Definition at line 180 of file `stl_vector.h`.

#### 5.738.2 Constructor & Destructor Documentation

**5.738.2.1** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>`  
`std::vector<_Tp, _Alloc>::vector ( ) [inline]`

Default constructor creates no elements.

Definition at line 217 of file `stl_vector.h`.

**5.738.2.2** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
std::vector<_Tp, _Alloc>::vector ( const allocator_type & __a )  
[inline, explicit]`

Creates a vector with no elements.

#### Parameters

*a* An allocator object.

Definition at line 225 of file `stl_vector.h`.

**5.738.2.3** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
std::vector<_Tp, _Alloc>::vector ( size_type __n ) [inline,  
explicit]`

Creates a vector with default constructed elements.

#### Parameters

*n* The number of elements to initially create.

This constructor fills the vector with *n* default constructed elements.

Definition at line 237 of file `stl_vector.h`.

**5.738.2.4** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
std::vector<_Tp, _Alloc>::vector ( size_type __n, const value_type  
& __value, const allocator_type & __a = allocator_type() )  
[inline]`

Creates a vector with copies of an exemplar element.

#### Parameters

*n* The number of elements to initially create.

*value* An element to copy.

*a* An allocator.

This constructor fills the vector with *n* copies of *value*.

Definition at line 249 of file `stl_vector.h`.

**5.738.2.5** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
std::vector<_Tp, _Alloc>::vector ( const vector<_Tp, _Alloc> &  
__x ) [inline]`

Vector copy constructor.

#### Parameters

*x* A vector of identical element and allocator types.

The newly-created vector uses a copy of the allocation object used by *x*. All the elements of *x* are copied, but any extra memory in *x* (for fast expansion) will not be copied.

Definition at line 278 of file `stl_vector.h`.

**5.738.2.6** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
std::vector<_Tp, _Alloc>::vector ( vector<_Tp, _Alloc> && __x  
) [inline]`

Vector move constructor.

#### Parameters

*x* A vector of identical element and allocator types.

The newly-created vector contains the exact contents of *x*. The contents of *x* are a valid, but unspecified vector.

Definition at line 294 of file `stl_vector.h`.

**5.738.2.7** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
std::vector<_Tp, _Alloc>::vector ( initializer_list<value_type  
> __l, const allocator_type & __a = allocator_type() )  
[inline]`

Builds a vector from an initializer list.

#### Parameters

*l* An [initializer\\_list](#).

*a* An allocator.

Create a vector consisting of copies of the elements in the [initializer\\_list](#) *l*.

This will call the element type's copy constructor *N* times (where *N* is *l.size()*) and do no memory reallocation.

Definition at line 308 of file `stl_vector.h`.

```
5.738.2.8 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
 template<typename _InputIterator > std::vector< _Tp, _Alloc
 >::vector (_InputIterator __first, _InputIterator __last, const
 allocator_type & __a = allocator_type()) [inline]
```

Builds a vector from a range.

#### Parameters

*first* An input iterator.

*last* An input iterator.

*a* An allocator.

Create a vector consisting of copies of the elements from [*first*,*last*).

If the iterators are forward, bidirectional, or random-access, then this will call the elements' copy constructor *N* times (where *N* is `distance(first,last)`) and do no memory reallocation. But if only input iterators are used, then this will do at most *2N* calls to the copy constructor, and  $\log N$  memory reallocations.

Definition at line 334 of file `stl_vector.h`.

```
5.738.2.9 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
 std::vector< _Tp, _Alloc >::~~vector () [inline]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 349 of file `stl_vector.h`.

**5.738.3 Member Function Documentation**

**5.738.3.1** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
 template<typename _ForwardIterator> pointer std::vector<_Tp,  
 _Alloc>::_M_allocate_and_copy ( size_type __n, _ForwardIterator  
 __first, _ForwardIterator __last ) [inline, protected]`

Memory expansion handler. Uses the member allocation function to obtain *n* bytes of memory, and then copies [first,last) into it.

Definition at line 1049 of file `std_vector.h`.

Referenced by `std::vector<_Tp, _Alloc>::operator=()`, and `std::vector<_Tp, _Alloc>::reserve()`.

**5.738.3.2** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
 void std::vector<_Tp, _Alloc>::_M_range_check ( size_type __n )  
 const [inline, protected]`

Safety check used only from `at()`.

Definition at line 716 of file `std_vector.h`.

Referenced by `std::vector< std::sub_match<_Bi_iter>, _Allocator>::at()`.

**5.738.3.3** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
 template<typename _InputIterator> void std::vector<_Tp,  
 _Alloc>::assign ( _InputIterator __first, _InputIterator __last )  
 [inline]`

Assigns a range to a vector.

**Parameters**

*first* An input iterator.

*last* An input iterator.

This function fills a vector with copies of the elements in the range [first,last).

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 429 of file `std_vector.h`.

**5.738.3.4** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
 void std::vector<_Tp, _Alloc>::assign ( size_type __n, const  
 value_type & __val ) [inline]`

Assigns a given value to a vector.

#### Parameters

*n* Number of elements to be assigned.

*val* Value to be assigned.

This function fills a vector with *n* copies of the given value. Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 412 of file `stl_vector.h`.

Referenced by `std::vector< std::sub_match<_Bi_iter>, _Allocator>::operator=()`.

**5.738.3.5** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
 void std::vector<_Tp, _Alloc>::assign ( initializer_list< value_type  
 > __l ) [inline]`

Assigns an initializer list to a vector.

#### Parameters

*l* An [initializer\\_list](#).

This function fills a vector with copies of the elements in the initializer list *l*.

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 449 of file `stl_vector.h`.

Referenced by `std::vector< std::sub_match<_Bi_iter>, _Allocator>::assign()`.

**5.738.3.6** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
 reference std::vector<_Tp, _Alloc>::at ( size_type __n )  
 [inline]`

Provides access to the data contained in the vector.

**Parameters**

*n* The index of the element for which data should be accessed.

**Returns**

Read/write reference to data.

**Exceptions**

*std::out\_of\_range* If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the vector. The function throws *out\_of\_range* if the check fails.

Definition at line 735 of file `stl_vector.h`.

```
5.738.3.7 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
 const_reference std::vector<_Tp, _Alloc>::at (size_type __n)
 const [inline]
```

Provides access to the data contained in the vector.

**Parameters**

*n* The index of the element for which data should be accessed.

**Returns**

Read-only (constant) reference to data.

**Exceptions**

*std::out\_of\_range* If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the vector. The function throws *out\_of\_range* if the check fails.

Definition at line 753 of file `stl_vector.h`.

```
5.738.3.8 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
 const_reference std::vector<_Tp, _Alloc>::back () const
 [inline]
```

Returns a read-only (constant) reference to the data at the last element of the vector.

Definition at line 788 of file `stl_vector.h`.

**5.738.3.9** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
reference std::vector<_Tp, _Alloc>::back ( ) [inline]`

Returns a read/write reference to the data at the last element of the vector.

Definition at line 780 of file `stl_vector.h`.

Referenced by `std::piecewise_linear_distribution<_RealType>::max()`, and `std::piecewise_constant_distribution<_RealType>::max()`.

**5.738.3.10** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_iterator std::vector<_Tp, _Alloc>::begin ( ) const  
[inline]`

Returns a read-only (constant) iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Reimplemented in `std::match_results<_Bi_iter, _Allocator>`, `std::match_results<_Bi_iter>`, and `std::match_results<_FwdIterT, _Alloc>`.

Definition at line 472 of file `stl_vector.h`.

**5.738.3.11** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
iterator std::vector<_Tp, _Alloc>::begin ( ) [inline]`

Returns a read/write iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Definition at line 463 of file `stl_vector.h`.

Referenced by `std::vector< std::sub_match<_Bi_iter>, _Allocator>::crend()`, `std::vector<_Tp, _Alloc>::emplace()`, `std::vector< std::sub_match<_Bi_iter>, _Allocator>::empty()`, `std::vector< std::sub_match<_Bi_iter>, _Allocator>::front()`, `std::vector<_Tp, _Alloc>::insert()`, `__gnu_parallel::multiseq_partition()`, `__gnu_parallel::multiseq_selection()`, `__gnu_parallel::multiway_merge_exact_splitting()`, `std::vector<_Tp, _Alloc>::operator=()`, `std::operator==()`, `std::vector< std::sub_match<_Bi_iter>, _Allocator>::rend()`, and `std::vector< std::sub_match<_Bi_iter>, _Allocator>::vector()`.

**5.738.3.12** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
size_type std::vector<_Tp, _Alloc>::capacity ( ) const  
[inline]`

Returns the total number of elements that the vector can hold before needing to allocate more memory.

Definition at line 650 of file `stl_vector.h`.



Referenced by `std::vector< _Tp, _Alloc >::operator=()`, and `std::vector< _Tp, _Alloc >::reserve()`.

**5.738.3.13** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_iterator std::vector< _Tp, _Alloc >::cbegin ( ) const  
[inline]`

Returns a read-only (constant) iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Reimplemented in `std::match_results< _Bi_iter, _Allocator >`, `std::match_results< _Bi_iter >`, and `std::match_results< _FwdIterT, _Alloc >`.

Definition at line 536 of file `stl_vector.h`.

**5.738.3.14** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_iterator std::vector< _Tp, _Alloc >::cend ( ) const  
[inline]`

Returns a read-only (constant) iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Reimplemented in `std::match_results< _Bi_iter, _Allocator >`, `std::match_results< _Bi_iter >`, and `std::match_results< _FwdIterT, _Alloc >`.

Definition at line 545 of file `stl_vector.h`.

**5.738.3.15** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::vector< _Tp, _Alloc >::clear ( ) [inline]`

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1039 of file `stl_vector.h`.

Referenced by `std::vector< std::sub_match< _Bi_iter >, _Allocator >::operator=()`.

**5.738.3.16** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_reverse_iterator std::vector< _Tp, _Alloc >::crbegin ( )  
const [inline]`

Returns a read-only (constant) reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

Definition at line 554 of file `stl_vector.h`.

**5.738.3.17** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_reverse_iterator std::vector< _Tp, _Alloc >::crend ( ) const  
[inline]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

Definition at line 563 of file `std_vector.h`.

**5.738.3.18** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
_Tp* std::vector< _Tp, _Alloc >::data ( ) [inline]`

Returns a pointer such that `[data(), data() + size())` is a valid range. For a non-empty vector, `data() == &front()`.

Definition at line 803 of file `std_vector.h`.

**5.738.3.19** `template<typename _Tp, typename _Alloc > template<typename...  
_Args> vector< _Tp, _Alloc >::iterator vector::emplace ( iterator  
__position, _Args &&... __args )`

Inserts an object in vector before specified iterator.

#### Parameters

*position* An iterator into the vector.

*args* Arguments.

#### Returns

An iterator that points to the inserted data.

This function will insert an object of type `T` constructed with `T(std::forward<Args>(args)...) before the specified location`. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

Definition at line 274 of file `vector.tcc`.

References `std::vector< _Tp, _Alloc >::begin()`, and `std::vector< _Tp, _Alloc >::end()`.

Referenced by `std::vector< std::sub_match< _Bi_iter >, _Allocator >::insert()`.

**5.738.3.20** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
bool std::vector<_Tp, _Alloc>::empty ( ) const [inline]`

Returns true if the vector is empty. (Thus `begin()` would equal `end()`.)

Reimplemented in `std::match_results<_Bi_iter, _Allocator>`, `std::match_results<_Bi_iter>`, and `std::match_results<_FwdIterT, _Alloc>`.

Definition at line 659 of file `stl_vector.h`.

Referenced by `std::piecewise_linear_distribution<_RealType>::densities()`, `std::piecewise_constant_distribution<_RealType>::densities()`, `std::piecewise_linear_distribution<_RealType>::intervals()`, `std::piecewise_constant_distribution<_RealType>::intervals()`, `std::piecewise_linear_distribution<_RealType>::max()`, `std::piecewise_constant_distribution<_RealType>::max()`, `std::discrete_distribution<_IntType>::max()`, `std::piecewise_linear_distribution<_RealType>::min()`, `std::piecewise_constant_distribution<_RealType>::min()`, and `std::discrete_distribution<_IntType>::probabilities()`.

**5.738.3.21** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
iterator std::vector<_Tp, _Alloc>::end ( ) [inline]`

Returns a read/write iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Definition at line 481 of file `stl_vector.h`.

Referenced by `std::vector<std::sub_match<_Bi_iter>, _Allocator>::back()`, `std::vector<std::sub_match<_Bi_iter>, _Allocator>::cbegin()`, `std::vector<_Tp, _Alloc>::emplace()`, `std::vector<std::sub_match<_Bi_iter>, _Allocator>::empty()`, `std::vector<_Tp, _Alloc>::erase()`, `std::vector<_Tp, _Alloc>::insert()`, `__gnu_parallel::multiseq_partition()`, `__gnu_parallel::multiseq_selection()`, `__gnu_parallel::multiway_merge_exact_splitting()`, `std::vector<_Tp, _Alloc>::operator=()`, `std::operator==()`, `std::vector<std::sub_match<_Bi_iter>, _Allocator>::push_back()`, `std::vector<std::sub_match<_Bi_iter>, _Allocator>::rbegin()`, `std::vector<std::sub_match<_Bi_iter>, _Allocator>::resize()`, and `std::vector<std::sub_match<_Bi_iter>, _Allocator>::vector()`.

**5.738.3.22** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_iterator std::vector<_Tp, _Alloc>::end ( ) const  
[inline]`

Returns a read-only (constant) iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Reimplemented in `std::match_results<_Bi_iter, _Allocator>`, `std::match_results<_Bi_iter>`, and `std::match_results<_FwdIterT, _Alloc>`.

Definition at line 490 of file `stl_vector.h`.

**5.738.3.23** `template<typename _Tp , typename _Alloc > vector< _Tp, _Alloc  
>::iterator vector::erase ( iterator __position )`

Remove element at given position.

#### Parameters

*position* Iterator pointing to element to be erased.

#### Returns

An iterator pointing to the next element (or `end()`).

This function will erase the element at the given position and thus shorten the vector by one.

Note This operation could be expensive and if it is frequently used the user should consider using `std::list`. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 136 of file `vector.tcc`.

References `std::vector< _Tp, _Alloc >::end()`.

**5.738.3.24** `template<typename _Tp , typename _Alloc > vector< _Tp, _Alloc  
>::iterator vector::erase ( iterator __first, iterator __last )`

Remove a range of elements.

#### Parameters

*first* Iterator pointing to the first element to be erased.

*last* Iterator pointing to one past the last element to be erased.

#### Returns

An iterator pointing to the element pointed to by *last* prior to erasing (or `end()`).

This function will erase the elements in the range `[first,last)` and shorten the vector accordingly.

Note This operation could be expensive and if it is frequently used the user should consider using [std::list](#). The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 148 of file `vector.tcc`.

References `std::vector<_Tp, _Alloc>::end()`.

#### 5.738.3.25 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::vector<_Tp, _Alloc>::front ( ) [inline]`

Returns a read/write reference to the data at the first element of the vector.

Definition at line 764 of file `stl_vector.h`.

Referenced by `std::vector< std::sub_match<_Bi_iter>, _Allocator>::data()`, `std::piecewise_linear_distribution<_RealType>::min()`, and `std::piecewise_constant_distribution<_RealType>::min()`.

#### 5.738.3.26 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::vector<_Tp, _Alloc>::front ( ) const [inline]`

Returns a read-only (constant) reference to the data at the first element of the vector.

Definition at line 772 of file `stl_vector.h`.

#### 5.738.3.27 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _InputIterator> void std::vector<_Tp, _Alloc>::insert ( iterator __position, _InputIterator __first, _InputIterator __last ) [inline]`

Inserts a range into the vector.

##### Parameters

*position* An iterator into the vector.

*first* An input iterator.

*last* An input iterator.

This function will insert copies of the data in the range `[first,last)` into the vector before the location specified by *pos*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using [std::list](#).

Definition at line 962 of file `stl_vector.h`.

```
5.738.3.28 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
 iterator std::vector<_Tp, _Alloc>::insert (iterator __position,
 value_type && __x) [inline]
```

Inserts given rvalue into vector before specified iterator.

#### Parameters

*position* An iterator into the vector.

*x* Data to be inserted.

#### Returns

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using [std::list](#).

Definition at line 908 of file `stl_vector.h`.

```
5.738.3.29 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
 void std::vector<_Tp, _Alloc>::insert (iterator __position,
 initializer_list< value_type > __l) [inline]
```

Inserts an [initializer\\_list](#) into the vector.

#### Parameters

*position* An iterator into the vector.

*l* An [initializer\\_list](#).

This function will insert copies of the data in the [initializer\\_list](#) *l* into the vector before the location specified by *position*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using [std::list](#).

Definition at line 925 of file `stl_vector.h`.

Referenced by `std::vector< std::sub_match<_Bi_iter>, _Allocator>::insert()`.

**5.738.3.30** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::vector< _Tp, _Alloc >::insert ( iterator __position,  
size_type __n, const value_type & __x ) [inline]`

Inserts a number of copies of given data into the vector.

#### Parameters

- position* An iterator into the vector.
- n* Number of elements to be inserted.
- x* Data to be inserted.

This function will insert a specified number of copies of the given data before the location specified by *position*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using [std::list](#).

Definition at line 943 of file `stl_vector.h`.

**5.738.3.31** `template<typename _Tp, typename _Alloc > vector< _Tp, _Alloc  
>::iterator vector::insert ( iterator __position, const value_type &  
__x )`

Inserts given value into vector before specified iterator.

#### Parameters

- position* An iterator into the vector.
- x* Data to be inserted.

#### Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using [std::list](#).

Definition at line 109 of file `vector.tcc`.

References `std::vector< _Tp, _Alloc >::begin()`, and `std::vector< _Tp, _Alloc >::end()`.

Referenced by `std::vector< std::sub_match< _Bi_iter >, _Allocator >::resize()`.

**5.738.3.32** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
size_type std::vector<_Tp, _Alloc>::max_size ( ) const  
[inline]`

Returns the [size\(\)](#) of the largest possible vector.

Reimplemented in `std::match_results<_Bi_iter, _Allocator>`, `std::match_results<_Bi_iter>`, and `std::match_results<_FwdIterT, _Alloc>`.

Definition at line 575 of file `stl_vector.h`.

Referenced by `std::vector<_Tp, _Alloc>::reserve()`.

**5.738.3.33** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
vector& std::vector<_Tp, _Alloc>::operator= ( vector<_Tp,  
_Alloc> && __x ) [inline]`

Vector move assignment operator.

#### Parameters

*x* A vector of identical element and allocator types.

The contents of *x* are moved into this vector (without copying). *x* is a valid, but unspecified vector.

Definition at line 373 of file `stl_vector.h`.

**5.738.3.34** `template<typename _Tp, typename _Alloc> vector<_Tp, _Alloc  
> & vector::operator= ( const vector<_Tp, _Alloc> & __x )`

Vector assignment operator.

#### Parameters

*x* A vector of identical element and allocator types.

All the elements of *x* are copied, but any extra memory in *x* (for fast expansion) will not be copied. Unlike the copy constructor, the allocator object is not copied.

Definition at line 159 of file `vector.tcc`.

References `std::_Destroy()`, `std::vector<_Tp, _Alloc>::_M_allocate_and_copy()`, `std::vector<_Tp, _Alloc>::begin()`, `std::vector<_Tp, _Alloc>::capacity()`, `std::vector<_Tp, _Alloc>::end()`, and `std::vector<_Tp, _Alloc>::size()`.



**5.738.3.35** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
vector& std::vector<_Tp, _Alloc>::operator= ( initializer_list<  
value_type > __l ) [inline]`

Vector list assignment operator.

#### Parameters

*l* An [initializer\\_list](#).

This function fills a vector with copies of the elements in the initializer list *l*.

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 394 of file `stl_vector.h`.

**5.738.3.36** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_reference std::vector<_Tp, _Alloc>::operator[] ( size_type  
__n ) const [inline]`

Subscript access to the data contained in the vector.

#### Parameters

*n* The index of the element for which data should be accessed.

#### Returns

Read-only (constant) reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and [out\\_of\\_range](#) lookups are not defined. (For checked lookups see [at\(\)](#).)

Definition at line 710 of file `stl_vector.h`.

**5.738.3.37** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
reference std::vector<_Tp, _Alloc>::operator[] ( size_type __n )  
[inline]`

Subscript access to the data contained in the vector.

**Parameters**

*n* The index of the element for which data should be accessed.

**Returns**

Read/write reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Definition at line 695 of file `stl_vector.h`.

**5.738.3.38** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::vector<_Tp, _Alloc>::pop_back ( ) [inline]`

Removes last element.

This is a typical stack operation. It shrinks the vector by one.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before `pop_back()` is called.

Definition at line 857 of file `stl_vector.h`.

**5.738.3.39** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::vector<_Tp, _Alloc>::push_back ( const value_type &  
__x ) [inline]`

Add data to the end of the vector.

**Parameters**

*x* Data to be added.

This is a typical stack operation. The function creates an element at the end of the vector and assigns the given data to it. Due to the nature of a vector this operation can be done in constant time if the vector has preallocated space available.

Definition at line 826 of file `stl_vector.h`.

Referenced by `__gnu_parallel::multiseq_partition()`, and `__gnu_parallel::multiseq_selection()`.

**5.738.3.40** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
reverse_iterator std::vector<_Tp, _Alloc>::rbegin ( )  
[inline]`

Returns a read/write reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

Definition at line 499 of file `stl_vector.h`.

**5.738.3.41** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_reverse_iterator std::vector<_Tp, _Alloc>::rbegin ( ) const  
[inline]`

Returns a read-only (constant) reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

Definition at line 508 of file `stl_vector.h`.

**5.738.3.42** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
reverse_iterator std::vector<_Tp, _Alloc>::rend ( ) [inline]`

Returns a read/write reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

Definition at line 517 of file `stl_vector.h`.

**5.738.3.43** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
const_reverse_iterator std::vector<_Tp, _Alloc>::rend ( ) const  
[inline]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

Definition at line 526 of file `stl_vector.h`.

**5.738.3.44** `template<typename _Tp, typename _Alloc> void vector::reserve (  
size_type __n )`

Attempt to preallocate enough memory for specified number of elements.

#### Parameters

*n* Number of elements required.

**Exceptions**

*`std::length_error`* If  $n$  exceeds `max_size()`.

This function attempts to reserve enough memory for the vector to hold the specified number of elements. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the number of elements that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of vector data.

Definition at line 67 of file `vector.tcc`.

References `std::_Destroy()`, `std::vector<_Tp, _Alloc>::_M_allocate_and_copy()`, `std::vector<_Tp, _Alloc>::capacity()`, `std::vector<_Tp, _Alloc>::max_size()`, and `std::vector<_Tp, _Alloc>::size()`.

**5.738.3.45** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::vector<_Tp, _Alloc>::resize ( size_type __new_size,  
const value_type & __x ) [inline]`

Resizes the vector to the specified number of elements.

**Parameters**

*`new_size`* Number of elements the vector should contain.

*`x`* Data with which new elements should be populated.

This function will resize the vector to the specified number of elements. If the number is smaller than the vector's current size the vector is truncated, otherwise the vector is extended and new elements are populated with given data.

Definition at line 609 of file `std_vector.h`.

**5.738.3.46** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::vector<_Tp, _Alloc>::resize ( size_type __new_size )  
[inline]`

Resizes the vector to the specified number of elements.

**Parameters**

*`new_size`* Number of elements the vector should contain.

This function will resize the vector to the specified number of elements. If the number is smaller than the vector's current size the vector is truncated, otherwise default constructed elements are appended.

Definition at line 589 of file `std_vector.h`.

Referenced by `__gnu_parallel::__shrink_and_double()`, `__gnu_parallel::multiway_merge_exact_splitting()`, and `__gnu_parallel::parallel_sort_mwms()`.

**5.738.3.47** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::vector<_Tp, _Alloc>::shrink_to_fit ( ) [inline]`

A non-binding request to reduce `capacity()` to `size()`.

Definition at line 641 of file `std_vector.h`.

**5.738.3.48** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
size_type std::vector<_Tp, _Alloc>::size ( ) const [inline]`

Returns the number of elements in the vector.

Reimplemented in `std::match_results<_Bi_iter, _Allocator>`, `std::match_results<_Bi_iter>`, and `std::match_results<_FwdIterT, _Alloc>`.

Definition at line 570 of file `std_vector.h`.

Referenced by `__gnu_parallel::__shrink()`, `__gnu_parallel::__shrink_and_double()`, `std::vector<std::sub_match<_Bi_iter>, _Allocator>::_M_range_check()`, `__gnu_parallel::list_partition()`, `std::discrete_distribution<_IntType>::max()`, `std::vector<_Tp, _Alloc>::operator=()`, `std::operator==()`, `std::vector<_Tp, _Alloc>::reserve()`, and `std::vector<std::sub_match<_Bi_iter>, _Allocator>::resize()`.

**5.738.3.49** `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>  
void std::vector<_Tp, _Alloc>::swap ( vector<_Tp, _Alloc> &  
__x ) [inline]`

Swaps data with another vector.

#### Parameters

*x* A vector of the same element and allocator types.

This exchanges the elements between two vectors in constant time. (Three pointers, so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(v1,v2)` will feed to this function.

Definition at line 1019 of file `stl_vector.h`.

Referenced by `std::vector< std::sub_match< _Bi_iter >, _Allocator >::operator=()`, `std::swap()`, and `std::vector< std::sub_match< _Bi_iter >, _Allocator >::swap()`.

The documentation for this class was generated from the following files:

- [stl\\_vector.h](#)
- [vector.tcc](#)

## 5.739 `std::vector< bool, _Alloc >` Class Template Reference

A specialization of `vector` for booleans which offers fixed time access to individual elements in any order.

Inherits `std::_Bvector_base< _Alloc >`.

### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_Bit_const_iterator` **const\_iterator**
- typedef `const bool *` **const\_pointer**
- typedef `bool` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `ptrdiff_t` **difference\_type**
- typedef `_Bit_iterator` **iterator**
- typedef `_Bit_reference *` **pointer**
- typedef `_Bit_reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `size_t` **size\_type**
- typedef `bool` **value\_type**

### Public Member Functions

- **vector** (`size_type __n`, `const bool &__value=bool()`, `const allocator_type &__a=allocator_type()`)
- `template<typename _InputIterator >`  
**vector** (`_InputIterator __first`, `_InputIterator __last`, `const allocator_type &__a=allocator_type()`)
- **vector** (`const vector &__x`)
- **vector** (`const allocator_type &__a`)
- **vector** (`vector &&__x`)
- **vector** (`initializer_list< bool > __l`, `const allocator_type &__a=allocator_type()`)

- void **assign** (size\_type \_\_n, const bool &\_\_x)
- template<typename \_InputIterator >  
void **assign** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void **assign** (initializer\_list< bool > \_\_l)
- reference **at** (size\_type \_\_n)
- const\_reference **at** (size\_type \_\_n) const
- reference **back** ()
- const\_reference **back** () const
- const\_iterator **begin** () const
- iterator **begin** ()
- size\_type **capacity** () const
- const\_iterator **cbegin** () const
- const\_iterator **cend** () const
- void **clear** ()
- const\_reverse\_iterator **crbegin** () const
- const\_reverse\_iterator **crend** () const
- void **data** ()
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- iterator **erase** (iterator \_\_position)
- iterator **erase** (iterator \_\_first, iterator \_\_last)
- void **flip** ()
- reference **front** ()
- const\_reference **front** () const
- allocator\_type **get\_allocator** () const
- void **insert** (iterator \_\_position, size\_type \_\_n, const bool &\_\_x)
- iterator **insert** (iterator \_\_position, const bool &\_\_x=bool())
- template<typename \_InputIterator >  
void **insert** (iterator \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- void **insert** (iterator \_\_p, initializer\_list< bool > \_\_l)
- size\_type **max\_size** () const
- vector & **operator=** (vector &&\_\_x)
- vector & **operator=** (const vector &\_\_x)
- vector & **operator=** (initializer\_list< bool > \_\_l)
- reference **operator[]** (size\_type \_\_n)
- const\_reference **operator[]** (size\_type \_\_n) const
- void **pop\_back** ()
- void **push\_back** (bool \_\_x)
- const\_reverse\_iterator **rbegin** () const
- reverse\_iterator **rbegin** ()
- reverse\_iterator **rend** ()
- const\_reverse\_iterator **rend** () const

- void **reserve** (size\_type \_\_n)
- void **resize** (size\_type \_\_new\_size, bool \_\_x=bool())
- void **shrink\_to\_fit** ()
- size\_type **size** () const
- void **swap** (vector &\_\_x)

### Static Public Member Functions

- static void **swap** (reference \_\_x, reference \_\_y)

### Protected Types

- typedef \_Alloc::template rebind< \_Bit\_type >::other **\_Bit\_alloc\_type**

### Protected Member Functions

- \_Bit\_type \* **\_M\_allocate** (size\_t \_\_n)
- template<typename \_InputIterator >  
void **\_M\_assign\_aux** (\_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
- template<typename \_ForwardIterator >  
void **\_M\_assign\_aux** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- template<typename \_Integer >  
void **\_M\_assign\_dispatch** (\_Integer \_\_n, \_Integer \_\_val, \_\_true\_type)
- template<class \_InputIterator >  
void **\_M\_assign\_dispatch** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- size\_type **\_M\_check\_len** (size\_type \_\_n, const char \*\_\_s) const
- iterator **\_M\_copy\_aligned** (const\_iterator \_\_first, const\_iterator \_\_last, iterator \_\_result)
- void **\_M\_deallocate** ()
- void **\_M\_erase\_at\_end** (iterator \_\_pos)
- void **\_M\_fill\_assign** (size\_t \_\_n, bool \_\_x)
- void **\_M\_fill\_insert** (iterator \_\_position, size\_type \_\_n, bool \_\_x)
- \_Bit\_alloc\_type & **\_M\_get\_Bit\_allocator** ()
- const \_Bit\_alloc\_type & **\_M\_get\_Bit\_allocator** () const
- void **\_M\_initialize** (size\_type \_\_n)
- template<typename \_InputIterator >  
void **\_M\_initialize\_dispatch** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- template<typename \_Integer >  
void **\_M\_initialize\_dispatch** (\_Integer \_\_n, \_Integer \_\_x, \_\_true\_type)



- `template<typename _InputIterator >`  
`void _M_initialize_range (_InputIterator __first, _InputIterator __last,`  
`std::input_iterator_tag)`
- `template<typename _ForwardIterator >`  
`void _M_initialize_range (_ForwardIterator __first, _ForwardIterator __last,`  
`std::forward_iterator_tag)`
- `void _M_insert_aux (iterator __position, bool __x)`
- `template<typename _Integer >`  
`void _M_insert_dispatch (iterator __pos, _Integer __n, _Integer __x, __true_`  
`type)`
- `template<typename _InputIterator >`  
`void _M_insert_dispatch (iterator __pos, _InputIterator __first, _InputIterator`  
`__last, __false_type)`
- `template<typename _ForwardIterator >`  
`void _M_insert_range (iterator __position, _ForwardIterator __first, _`  
`ForwardIterator __last, std::forward_iterator_tag)`
- `template<typename _InputIterator >`  
`void _M_insert_range (iterator __pos, _InputIterator __first, _InputIterator __`  
`last, std::input_iterator_tag)`
- `void _M_range_check (size_type __n) const`

### Protected Attributes

- `_Bvector_impl _M_impl`

### Friends

- class `hash`

#### 5.739.1 Detailed Description

`template<typename _Alloc> class std::vector< bool, _Alloc >`

A specialization of `vector` for booleans which offers fixed time access to individual elements in any order. Note that `vector<bool>` does not actually meet the requirements for being a container. This is because the reference and pointer types are not really references and pointers to `bool`. See DR96 for details.

### See also

[vector](#) for function documentation.

In some terminology a vector can be described as a dynamic C-style array, it offers fast and efficient access to individual elements in any order and saves the user from

worrying about memory and size allocation. Subscripting ( `[]` ) access is also provided as with C-style arrays.

Definition at line 479 of file `std_bvector.h`.

The documentation for this class was generated from the following files:

- [std\\_bvector.h](#)
- [vector.tcc](#)

## 5.740 `std::weak_ptr<_Tp>` Class Template Reference

A smart pointer with weak semantics.

Inherits `std::__weak_ptr<_Tp>`.

### Public Types

- `typedef _Tp element_type`

### Public Member Functions

- `template<typename _Tp1, typename = typename std::enable_if<std::is_convertible<_Tp1*, __weak_ptr*>::value>::type>`  
`weak_ptr (const weak\_ptr<_Tp1> &__r)`
- `template<typename _Tp1, typename = typename std::enable_if<std::is_convertible<_Tp1*, __shared_ptr*>::value>::type>`  
`weak_ptr (const shared\_ptr<_Tp1> &__r)`
- `bool expired () const`
- `shared\_ptr<_Tp> lock () const`
- `template<typename _Tp1>`  
`weak\_ptr & operator= (const weak\_ptr<_Tp1> &__r)`
- `template<typename _Tp1>`  
`weak\_ptr & operator= (const shared\_ptr<_Tp1> &__r)`
- `template<typename _Tp1>`  
`bool owner_before (const __weak_ptr<_Tp1, _Lp> &__rhs) const`
- `template<typename _Tp1>`  
`bool owner_before (const __shared_ptr<_Tp1, _Lp> &__rhs) const`
- `void reset ()`
- `void swap (__weak_ptr &__s)`
- `long use_count () const`

### 5.740.1 Detailed Description

**template<typename \_Tp> class std::weak\_ptr< \_Tp >**

A smart pointer with weak semantics. With forwarding constructors and assignment operators.

Definition at line 394 of file shared\_ptr.h.

The documentation for this class was generated from the following file:

- [shared\\_ptr.h](#)

## 5.741 std::weibull\_distribution< \_RealType > Class Template Reference

A [weibull\\_distribution](#) random number distribution.

### Classes

- struct [param\\_type](#)

### Public Types

- typedef \_RealType [result\\_type](#)

### Public Member Functions

- **weibull\_distribution** (\_RealType \_\_a=\_RealType(1), \_RealType \_\_b=\_RealType(1))
- **weibull\_distribution** (const [param\\_type](#) &\_\_p)
- \_RealType **a** () const
- \_RealType **b** () const
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- void **param** (const [param\\_type](#) &\_\_param)
- [param\\_type](#) **param** () const
- void **reset** ()

### 5.741.1 Detailed Description

`template<typename _RealType = double> class std::weibull_distribution< _RealType >`

A [weibull\\_distribution](#) random number distribution. The formula for the normal probability density function is:

$$p(x|\alpha, \beta) = \frac{\alpha}{\beta} \left(\frac{x}{\beta}\right)^{\alpha-1} \exp\left(-\left(\frac{x}{\beta}\right)^\alpha\right)$$

Definition at line 4312 of file random.h.

### 5.741.2 Member Typedef Documentation

**5.741.2.1** `template<typename _RealType = double> typedef _RealType  
std::weibull_distribution< _RealType >::result_type`

The type of the range of the distribution.

Definition at line 4319 of file random.h.

### 5.741.3 Member Function Documentation

**5.741.3.1** `template<typename _RealType = double> _RealType  
std::weibull_distribution< _RealType >::a ( ) const [inline]`

Return the  $a$  parameter of the distribution.

Definition at line 4370 of file random.h.

**5.741.3.2** `template<typename _RealType = double> _RealType  
std::weibull_distribution< _RealType >::b ( ) const [inline]`

Return the  $b$  parameter of the distribution.

Definition at line 4377 of file random.h.

**5.741.3.3** `template<typename _RealType = double> result_type  
std::weibull_distribution< _RealType >::max ( ) const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 4406 of file random.h.

**5.741.3.4** `template<typename _RealType = double> result_type  
std::weibull_distribution< _RealType >::min ( ) const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 4399 of file random.h.

**5.741.3.5** `template<typename _RealType = double> template<typename  
_UniformRandomNumberGenerator > result_type  
std::weibull_distribution< _RealType >::operator() (   
_UniformRandomNumberGenerator & __urng ) [inline]`

Generating functions.

Definition at line 4414 of file random.h.

References `std::weibull_distribution< _RealType >::operator()()`, and `std::weibull_distribution< _RealType >::param()`.

Referenced by `std::weibull_distribution< _RealType >::operator()()`.

**5.741.3.6** `template<typename _RealType = double> void  
std::weibull_distribution< _RealType >::param ( const param_type  
& __param ) [inline]`

Sets the parameter set of the distribution.

#### Parameters

***\_\_param*** The new parameter set of the distribution.

Definition at line 4392 of file random.h.

**5.741.3.7** `template<typename _RealType = double> param_type  
std::weibull_distribution< _RealType >::param ( ) const  
[inline]`

## 5.742 `std::weibull_distribution<_RealType>::param_type` Struct Reference 3403

Returns the parameter set of the distribution.

Definition at line 4384 of file `random.h`.

Referenced by `std::weibull_distribution<_RealType>::operator()()`, `std::operator==( )`, and `std::operator>( )`.

### 5.741.3.8 `template<typename _RealType = double> void std::weibull_distribution<_RealType>::reset( ) [inline]`

Resets the distribution state.

Definition at line 4363 of file `random.h`.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

## 5.742 `std::weibull_distribution<_RealType>::param_type` Struct Reference

### Public Types

- typedef [weibull\\_distribution<\\_RealType>](#) **distribution\_type**

### Public Member Functions

- **param\_type** (`_RealType __a=_RealType(1), _RealType __b=_RealType(1)`)
- `_RealType a` () const
- `_RealType b` () const

### Friends

- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

### 5.742.1 Detailed Description

`template<typename _RealType = double> struct std::weibull_distribution<_RealType>::param_type`

Parameter type.

Definition at line 4321 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 6 File Documentation

### 6.1 algo.h File Reference

Parallel STL function calls corresponding to the [stl\\_algo.h](#) header.

#### Classes

- struct [std::\\_\\_parallel::\\_\\_CRandNumber< \\_MustBeInt >](#)  
*Functor wrapper for std::rand().*

#### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_parallel](#)

#### Functions

- template<typename \_RAIter >  
\_RAIter **std::\_\_parallel::\_\_adjacent\_find\_switch** (\_RAIter \_\_begin, \_RAIter \_\_end, random\_access\_iterator\_tag)
- template<typename \_FIterator, typename \_IteratorTag >  
\_FIterator **std::\_\_parallel::\_\_adjacent\_find\_switch** (\_FIterator \_\_begin, \_FIterator \_\_end, \_IteratorTag)
- template<typename \_FIterator, typename \_BinaryPredicate, typename \_IteratorTag >  
\_FIterator **std::\_\_parallel::\_\_adjacent\_find\_switch** (\_FIterator \_\_begin, \_FIterator \_\_end, \_BinaryPredicate \_\_pred, \_IteratorTag)
- template<typename \_RAIter, typename \_BinaryPredicate >  
\_RAIter **std::\_\_parallel::\_\_adjacent\_find\_switch** (\_RAIter \_\_begin, \_RAIter \_\_end, \_BinaryPredicate \_\_pred, random\_access\_iterator\_tag)
- template<typename \_RAIter, typename \_Predicate >  
iterator\_traits< \_RAIter >::difference\_type **std::\_\_parallel::\_\_count\_if\_switch** (\_RAIter \_\_begin, \_RAIter \_\_end, \_Predicate \_\_pred, random\_access\_iterator\_tag, [\\_\\_gnu\\_parallel::Parallelism](#) \_\_parallelism\_tag=\_\_gnu\_parallel::parallel\_unbalanced)

- `template<typename _Iter, typename _Predicate, typename _IteratorTag >`  
`iterator_traits< _Iter >::difference_type std::__parallel::__count_if_switch`  
`( _Iter __begin, _Iter __end, _Predicate __pred, _IteratorTag)`
- `template<typename _RAIter, typename _Tp >`  
`iterator_traits< _RAIter >::difference_type std::__parallel::__count_`  
`switch ( _RAIter __begin, _RAIter __end, const _Tp &__value, random_`  
`access_iterator_tag, \_\_gnu\_parallel::\_\_Parallelism __parallelism_tag=__gnu_`  
`parallel::parallel_unbalanced)`
- `template<typename _Iter, typename _Tp, typename _IteratorTag >`  
`iterator_traits< _Iter >::difference_type std::__parallel::__count_switch (`  
`_Iter __begin, _Iter __end, const _Tp &__value, _IteratorTag)`
- `template<typename _RAIter, typename _FIterator, typename _BinaryPredicate, typename _`  
`IteratorTag >`  
`_RAIter std::__parallel::__find_first_of_switch ( _RAIter __begin1, _RAIter`  
`__end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp,`  
`random_access_iterator_tag, _IteratorTag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate, typename _`  
`IteratorTag1, typename _IteratorTag2 >`  
`_Iter std::__parallel::__find_first_of_switch ( _Iter __begin1, _Iter __`  
`end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, _`  
`IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _FIterator, typename _IteratorTag1, typename _IteratorTag2`  
`>`  
`_Iter std::__parallel::__find_first_of_switch ( _Iter __begin1, _Iter __end1,`  
`_FIterator __begin2, _FIterator __end2, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _Predicate, typename _IteratorTag >`  
`_Iter std::__parallel::__find_if_switch ( _Iter __begin, _Iter __end, _`  
`Predicate __pred, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate >`  
`_RAIter std::__parallel::__find_if_switch ( _RAIter __begin, _RAIter __end,`  
`_Predicate __pred, random_access_iterator_tag)`
- `template<typename _Iter, typename _Tp, typename _IteratorTag >`  
`_Iter std::__parallel::__find_switch ( _Iter __begin, _Iter __end, const _Tp`  
`&__val, _IteratorTag)`
- `template<typename _RAIter, typename _Tp >`  
`_RAIter std::__parallel::__find_switch ( _RAIter __begin, _RAIter __end,`  
`const _Tp &__val, random_access_iterator_tag)`
- `template<typename _Iter, typename _Function, typename _IteratorTag >`  
`_Function std::__parallel::__for_each_switch ( _Iter __begin, _Iter __end, _`  
`Function __f, _IteratorTag)`
- `template<typename _RAIter, typename _Function >`  
`_Function std::__parallel::__for_each_switch ( _RAIter __begin, _RAIter _`  
`__end, _Function __f, random_access_iterator_tag, \_\_gnu\_parallel::\_\_Parallelism`  
`__parallelism_tag=__gnu_parallel::parallel_balanced)`



- `template<typename _OutputIterator , typename _Size , typename _Generator , typename _IteratorTag >`  
`_OutputIterator std::__parallel::__generate_n_switch (_OutputIterator __begin, _Size __n, _Generator __gen, _IteratorTag)`
- `template<typename _RAIter , typename _Size , typename _Generator >`  
`_RAIter std::__parallel::__generate_n_switch (_RAIter __begin, _Size __n, _Generator __gen, random_access_iterator_tag, \_\_gnu\_parallel::\_\_Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _FIterator , typename _Generator , typename _IteratorTag >`  
`void std::__parallel::__generate_switch (_FIterator __begin, _FIterator __end, _Generator __gen, _IteratorTag)`
- `template<typename _RAIter , typename _Generator >`  
`void std::__parallel::__generate_switch (_RAIter __begin, _RAIter __end, _Generator __gen, random_access_iterator_tag, \_\_gnu\_parallel::\_\_Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _FIterator , typename _Compare , typename _IteratorTag >`  
`_FIterator std::__parallel::__max_element_switch (_FIterator __begin, _FIterator __end, _Compare __comp, _IteratorTag)`
- `template<typename _RAIter , typename _Compare >`  
`_RAIter std::__parallel::__max_element_switch (_RAIter __begin, _RAIter __end, _Compare __comp, random_access_iterator_tag, \_\_gnu\_parallel::\_\_Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _Iter1 , typename _Iter2 , typename _OutputIterator , typename _Compare , typename _IteratorTag1 , typename _IteratorTag2 , typename _IteratorTag3 >`  
`_OutputIterator std::__parallel::__merge_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Compare __comp, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _Iter1 , typename _Iter2 , typename _OutputIterator , typename _Compare >`  
`_OutputIterator std::__parallel::__merge_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Compare __comp, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _FIterator , typename _Compare , typename _IteratorTag >`  
`_FIterator std::__parallel::__min_element_switch (_FIterator __begin, _FIterator __end, _Compare __comp, _IteratorTag)`
- `template<typename _RAIter , typename _Compare >`  
`_RAIter std::__parallel::__min_element_switch (_RAIter __begin, _RAIter __end, _Compare __comp, random_access_iterator_tag, \_\_gnu\_parallel::\_\_Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _FIterator , typename _Predicate , typename _IteratorTag >`  
`_FIterator std::__parallel::__partition_switch (_FIterator __begin, _FIterator __end, _Predicate __pred, _IteratorTag)`
- `template<typename _RAIter , typename _Predicate >`  
`_RAIter std::__parallel::__partition_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)`

- `template<typename _FIterator, typename _Predicate, typename _Tp, typename _IteratorTag >`  
`void std::__parallel::__replace_if_switch (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate, typename _Tp >`  
`void std::__parallel::__replace_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, const _Tp &__new_value, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _FIterator, typename _Tp, typename _IteratorTag >`  
`void std::__parallel::__replace_switch (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value, _IteratorTag)`
- `template<typename _RAIter, typename _Tp >`  
`void std::__parallel::__replace_switch (_RAIter __begin, _RAIter __end, const _Tp &__old_value, const _Tp &__new_value, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate, typename _IteratorTag >`  
`_FIterator std::__parallel::__search_n_switch (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, _IteratorTag)`
- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BinaryPredicate >`  
`_RAIter std::__parallel::__search_n_switch (_RAIter __begin, _RAIter __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, random_access_iterator_tag)`
- `template<typename _RAIter1, typename _RAIter2 >`  
`_RAIter1 std::__parallel::__search_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _FIterator1, typename _FIterator2, typename _IteratorTag1, typename _IteratorTag2 >`  
`_FIterator1 std::__parallel::__search_switch (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _BinaryPredicate >`  
`_RAIter1 std::__parallel::__search_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _BinaryPredicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2 >`  
`_FIterator1 std::__parallel::__search_switch (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`

- ```

_OutputIterator std::__parallel::__set_difference_switch (_Iter1 __begin1, _
_Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _
Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)

```
- ```

template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename
_Predicate >
_Output_RAlter std::__parallel::__set_difference_switch (_RAIter1 __
__begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _
Output_RAlter __result, _Predicate __pred, random_access_iterator_tag,
random_access_iterator_tag, random_access_iterator_tag)

```
  - ```

template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator,
typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >
_OutputIterator std::__parallel::__set_intersection_switch (_Iter1 __begin1,
_Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _
Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)

```
 - ```

template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _
Predicate >
_Output_RAlter std::__parallel::__set_intersection_switch (_RAIter1 __
begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_
RAIter __result, _Predicate __pred, random_access_iterator_tag, random_
access_iterator_tag, random_access_iterator_tag)

```
  - ```

template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator,
typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >
_OutputIterator std::__parallel::__set_symmetric_difference_switch (_Iter1
__begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __
result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)

```
 - ```

template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename
_Predicate >
_Output_RAlter std::__parallel::__set_symmetric_difference_switch (_
RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2,
_Output_RAlter __result, _Predicate __pred, random_access_iterator_tag,
random_access_iterator_tag, random_access_iterator_tag)

```
  - ```

template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator,
typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >
_OutputIterator std::__parallel::__set_union_switch (_Iter1 __begin1, _Iter1
__end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Predicate
__pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)

```
 - ```

template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _
Predicate >
_Output_RAlter std::__parallel::__set_union_switch (_RAIter1 __begin1, _
RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAlter
__result, _Predicate __pred, random_access_iterator_tag, random_access_
iterator_tag, random_access_iterator_tag)

```
  - ```

template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation >
_RAIter2 std::__parallel::__transform1_switch (_RAIter1 __begin, _RAIter1

```

- ```

__end, _RAIter2 __result, _UnaryOperation __unary_op, random_access_
iterator_tag, random_access_iterator_tag, __gnu_parallel::__Parallelism __
parallelism_tag=__gnu_parallel::parallel_balanced)

```
- `template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation, typename _IteratorTag1, typename _IteratorTag2 >`  
`_RAIter2 std::__parallel::__transform1_switch (_RAIter1 __begin, _RAIter1`  
`__end, _RAIter2 __result, _UnaryOperation __unary_op, _IteratorTag1, _`  
`IteratorTag2)`
  - `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BinaryOperation >`  
`_RAIter3 std::__parallel::__transform2_switch (_RAIter1 __begin1, _`  
`RAIter1 __end1, _RAIter2 __begin2, _RAIter3 __result, _BinaryOperation __`  
`binary_op, random_access_iterator_tag, random_access_iterator_tag, random_`  
`access_iterator_tag, \_\_gnu\_parallel::\_\_Parallelism __parallelism_tag=__gnu_`  
`parallel::parallel_balanced)`
  - `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation, typename _Tag1, typename _Tag2, typename _Tag3 >`  
`_OutputIterator std::__parallel::__transform2_switch (_Iter1 __begin1, _`  
`Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __`  
`binary_op, _Tag1, _Tag2, _Tag3)`
  - `template<typename _Iter, typename _OutputIterator, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >`  
`_OutputIterator std::__parallel::__unique_copy_switch (_Iter __begin, _Iter`  
`__last, _OutputIterator __out, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
  - `template<typename _RAIter, typename RandomAccessOutputIterator, typename _Predicate >`  
`RandomAccessOutputIterator std::__parallel::__unique_copy_switch (_`  
`RAIter __begin, _RAIter __last, RandomAccessOutputIterator __out, _`  
`Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
  - `template<typename _FIterator >`  
`_FIterator std::__parallel::adjacent_find (_FIterator __begin, _FIterator __`  
`end, \_\_gnu\_parallel::sequential\_tag)`
  - `template<typename _FIterator, typename _BinaryPredicate >`  
`_FIterator std::__parallel::adjacent_find (_FIterator __begin, _FIterator __`  
`end, _BinaryPredicate __binary_pred, \_\_gnu\_parallel::sequential\_tag)`
  - `template<typename _FIterator >`  
`_FIterator std::__parallel::adjacent_find (_FIterator __begin, _FIterator __`  
`end)`
  - `template<typename _FIterator, typename _BinaryPredicate >`  
`_FIterator std::__parallel::adjacent_find (_FIterator __begin, _FIterator __`  
`end, _BinaryPredicate __pred)`
  - `template<typename _Iter, typename _Tp >`  
`iterator_traits< _Iter >::difference_type std::__parallel::count (_Iter __`  
`begin, _Iter __end, const _Tp &__value, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _Iter, typename _Tp >`  
`iterator_traits< _Iter >::difference_type std::__parallel::count ( _Iter __begin, _Iter __end, const _Tp &__value, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp >`  
`iterator_traits< _Iter >::difference_type std::__parallel::count ( _Iter __begin, _Iter __end, const _Tp &__value)`
- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >::difference_type std::__parallel::count_if ( _Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >::difference_type std::__parallel::count_if ( _Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >::difference_type std::__parallel::count_if ( _Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Tp >`  
`_Iter std::__parallel::find ( _Iter __begin, _Iter __end, const _Tp &__val)`
- `template<typename _Iter, typename _Tp >`  
`_Iter std::__parallel::find ( _Iter __begin, _Iter __end, const _Tp &__val, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate >`  
`_Iter std::__parallel::find_first_of ( _Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp)`
- `template<typename _Iter, typename _FIterator >`  
`_Iter std::__parallel::find_first_of ( _Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2)`
- `template<typename _Iter, typename _FIterator >`  
`_Iter std::__parallel::find_first_of ( _Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate >`  
`_Iter std::__parallel::find_first_of ( _Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter std::__parallel::find_if ( _Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter std::__parallel::find_if ( _Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iterator, typename _Function >`  
`_Function std::__parallel::for_each ( _Iterator __begin, _Iterator __end, _Function __f, \_\_gnu\_parallel::Parallelism __parallelism_tag)`

- `template<typename _Iter, typename _Function >`  
`_Function std::__parallel::for_each (_Iter __begin, _Iter __end, _Function`  
`__f, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iterator, typename _Function >`  
`_Function std::__parallel::for_each (_Iterator __begin, _Iterator __end, _`  
`Function __f)`
- `template<typename _FIterator, typename _Generator >`  
`void std::__parallel::generate (_FIterator __begin, _FIterator __end, _`  
`Generator __gen, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Generator >`  
`void std::__parallel::generate (_FIterator __begin, _FIterator __end, _`  
`Generator __gen, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Generator >`  
`void std::__parallel::generate (_FIterator __begin, _FIterator __end, _`  
`Generator __gen)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`_OutputIterator std::__parallel::generate_n (_OutputIterator __begin, _Size _`  
`_n, _Generator __gen, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`_OutputIterator std::__parallel::generate_n (_OutputIterator __begin, _Size _`  
`_n, _Generator __gen, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`_OutputIterator std::__parallel::generate_n (_OutputIterator __begin, _Size _`  
`_n, _Generator __gen)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator std::__parallel::max_element (_FIterator __begin, _FIterator __end,`  
`_Compare __comp, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator >`  
`_FIterator std::__parallel::max_element (_FIterator __begin, _FIterator __end,`  
`\_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator >`  
`_FIterator std::__parallel::max_element (_FIterator __begin, _FIterator __`  
`end)`
- `template<typename _FIterator >`  
`_FIterator std::__parallel::max_element (_FIterator __begin, _FIterator __end,`  
`\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator std::__parallel::max_element (_FIterator __begin, _FIterator __end,`  
`_Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator std::__parallel::max_element (_FIterator __begin, _FIterator __end,`  
`_Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`  
`_OutputIterator std::__parallel::merge (_Iter1 __begin1, _Iter1 __`  
`end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, \_\_gnu\_`  
`parallel::sequential\_tag)`

- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator std::__parallel::merge (_Iter1 __begin1, _Iter1 __end1, _-`  
`_Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Compare __comp,`  
`\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`  
`_OutputIterator std::__parallel::merge (_Iter1 __begin1, _Iter1 __end1, _-`  
`_Iter2 __begin2, _Iter2 __end2, _OutputIterator __result)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator std::__parallel::merge (_Iter1 __begin1, _Iter1 __end1, _-`  
`_Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _FIterator >`  
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end,`  
`\_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end,`  
`_Compare __comp)`
- `template<typename _FIterator >`  
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end,`  
`\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end,`  
`_Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator >`  
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end,`  
`_Compare __comp, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter`  
`__end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter`  
`__end, _Compare __comp)`
- `template<typename _RAIter >`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter`  
`__end)`
- `template<typename _RAIter >`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter`  
`__end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _-`  
`_RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _RAIter >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _-`  
`RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _-`  
`RAIter __end)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _-`  
`RAIter __end, _Compare __comp)`
- `template<typename _FIterator, typename _Predicate >`  
`_FIterator std::__parallel::partition (_FIterator __begin, _FIterator __end, _-`  
`Predicate __pred)`
- `template<typename _FIterator, typename _Predicate >`  
`_FIterator std::__parallel::partition (_FIterator __begin, _FIterator __end, _-`  
`Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter >`  
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, \_\_-`  
`gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, _-`  
`RandomNumberGenerator &__rand, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, _-`  
`RandomNumberGenerator &&__rand)`
- `template<typename _FIterator, typename _Tp >`  
`void std::__parallel::replace (_FIterator __begin, _FIterator __end, const _Tp`  
`&__old_value, const _Tp &__new_value, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Tp >`  
`void std::__parallel::replace (_FIterator __begin, _FIterator __end, const _-`  
`Tp &__old_value, const _Tp &__new_value, \_\_gnu\_parallel::\_Parallelism __-`  
`parallelism_tag)`
- `template<typename _FIterator, typename _Tp >`  
`void std::__parallel::replace (_FIterator __begin, _FIterator __end, const _Tp`  
`&__old_value, const _Tp &__new_value)`
- `template<typename _FIterator, typename _Predicate, typename _Tp >`  
`void std::__parallel::replace_if (_FIterator __begin, _FIterator __end, _-`  
`Predicate __pred, const _Tp &__new_value, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Predicate, typename _Tp >`  
`void std::__parallel::replace_if (_FIterator __begin, _FIterator __end, _-`  
`Predicate __pred, const _Tp &__new_value, \_\_gnu\_parallel::\_Parallelism __-`  
`parallelism_tag)`



- `template<typename _FIterator, typename _Predicate, typename _Tp >`  
`void std::__parallel::replace_if (_FIterator __begin, _FIterator __end, _-`  
`Predicate __pred, const _Tp &__new_value)`
- `template<typename _FIterator1, typename _FIterator2 >`  
`_FIterator1 std::__parallel::search (_FIterator1 __begin1, _FIterator1 __end1,`  
`_FIterator2 __begin2, _FIterator2 __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate >`  
`_FIterator1 std::__parallel::search (_FIterator1 __begin1, _FIterator1 __end1,`  
`_FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred)`
- `template<typename _FIterator1, typename _FIterator2 >`  
`_FIterator1 std::__parallel::search (_FIterator1 __begin1, _FIterator1 __end1,`  
`_FIterator2 __begin2, _FIterator2 __end2)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate >`  
`_FIterator1 std::__parallel::search (_FIterator1 __begin1, _FIterator1 __end1,`  
`_FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred, \_\_gnu-`  
`parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp >`  
`_FIterator std::__parallel::search_n (_FIterator __begin, _FIterator __end, _-`  
`Integer __count, const _Tp &__val, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`  
`_FIterator std::__parallel::search_n (_FIterator __begin, _FIterator __end, _-`  
`Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, \_\_gnu-`  
`parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`  
`_FIterator std::__parallel::search_n (_FIterator __begin, _FIterator __end, _-`  
`Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)`
- `template<typename _FIterator, typename _Integer, typename _Tp >`  
`_FIterator std::__parallel::search_n (_FIterator __begin, _FIterator __end, _-`  
`Integer __count, const _Tp &__val)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::__parallel::set_difference (_IIter1 __begin1, _IIter1 __-`  
`end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __-`  
`pred)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`  
`_OutputIterator std::__parallel::set_difference (_IIter1 __begin1, _IIter1 __-`  
`end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, \_\_gnu-`  
`parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::__parallel::set_difference (_IIter1 __begin1, _IIter1 __-`  
`end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __-`  
`pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`  
`_OutputIterator std::__parallel::set_difference (_IIter1 __begin1, _IIter1 __-`  
`end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out)`

- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`  
`_OutputIterator std::__parallel::set_intersection (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::__parallel::set_intersection (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::__parallel::set_intersection (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`  
`_OutputIterator std::__parallel::set_intersection (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::__parallel::set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`  
`_OutputIterator std::__parallel::set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::__parallel::set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`  
`_OutputIterator std::__parallel::set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::__parallel::set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::__parallel::set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`  
`_OutputIterator std::__parallel::set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`  
`_OutputIterator std::__parallel::set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out)`

- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::default\_parallel\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::parallel\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::balanced\_quicksort\_tag __parallelism)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_exact\_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::quicksort\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_sampling\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)`

- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::quicksort\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::default\_parallel\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::parallel\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::balanced\_quicksort\_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation >`  
`_OutputIterator std::__parallel::transform (_Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::__parallel::transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation >`  
`_OutputIterator std::__parallel::transform (_Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::__parallel::transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation >`  
`_OutputIterator std::__parallel::transform (_Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op, \_\_gnu\_parallel::\_\_Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::__parallel::transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, \_\_gnu\_parallel::\_\_Parallelism __parallelism_tag)`

- `template<typename _Iter , typename _OutputIterator >`  
`_OutputIterator std::__parallel::unique_copy (_Iter __begin1, _Iter __end1,`  
`_OutputIterator __out)`
- `template<typename _Iter , typename _OutputIterator , typename _Predicate >`  
`_OutputIterator std::__parallel::unique_copy (_Iter __begin1, _Iter __end1,`  
`_OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter , typename _OutputIterator >`  
`_OutputIterator std::__parallel::unique_copy (_Iter __begin1, _Iter __end1,`  
`_OutputIterator __out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter , typename _OutputIterator , typename _Predicate >`  
`_OutputIterator std::__parallel::unique_copy (_Iter __begin1, _Iter __end1,`  
`_OutputIterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`

### 6.1.1 Detailed Description

Parallel STL function calls corresponding to the [stl\\_algo.h](#) header. The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [algo.h](#).

## 6.2 `algbase.h` File Reference

Parallel STL function calls corresponding to the [stl\\_algbase.h](#) header. The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

### Namespaces

- namespace [std](#)
- namespace `std::__parallel`

### Functions

- `template<typename _Iter1 , typename _Iter2 , typename _Predicate , typename _IteratorTag1 ,`  
`typename _IteratorTag2 >`  
`bool std::__parallel::__lexicographical_compare_switch (_Iter1 __begin1,`  
`_Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, _`  
`IteratorTag1, _IteratorTag2)`

- `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`  
`bool std::__parallel::__lexicographical_compare_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`  
`pair< _RAIter1, _RAIter2 > std::__parallel::__mismatch_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >`  
`pair< _Iter1, _Iter2 > std::__parallel::__mismatch_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`

- `template<typename _Iter1, typename _Iter2 >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1`  
`__end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2 >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1`  
`__end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag)`

### 6.2.1 Detailed Description

Parallel STL function calls corresponding to the [stl\\_algobase.h](#) header. The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [algobase.h](#).

## 6.3 algorithm File Reference

### 6.3.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [algorithm](#).

## 6.4 algorithm File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Functions

- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`  
`pair< _InputIterator, _OutputIterator > \_\_gnu\_cxx::\_\_copy\_n (_InputIterator`  
`__first, _Size __count, _OutputIterator __result, input_iterator_tag)`
- `template<typename _RAIterator, typename _Size, typename _OutputIterator >`  
`pair< _RAIterator, _OutputIterator > \_\_gnu\_cxx::\_\_copy\_n (_RAIterator __-`  
`first, _Size __count, _OutputIterator __result, random_access_iterator_tag)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`int \_\_gnu\_cxx::\_\_lexicographical\_compare\_3way (_InputIterator1 __first1,`  
`_InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `int \_\_gnu\_cxx::\_\_lexicographical\_compare\_3way (const unsigned char *__-`  
`first1, const unsigned char *__last1, const unsigned char *__first2, const un-`  
`signed char *__last2)`

- `int __gnu_cxx::__lexicographical_compare_3way (const char *__first1, const char *__last1, const char *__first2, const char *__last2)`
- `template<typename _Tp, typename _Compare >`  
`const _Tp & __gnu_cxx::__median (const _Tp &__a, const _Tp &__b, const _Tp &__c, _Compare __comp)`
- `template<typename _Tp >`  
`const _Tp & __gnu_cxx::__median (const _Tp &__a, const _Tp &__b, const _Tp &__c)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Distance >`  
`_RandomAccessIterator __gnu_cxx::__random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out, const _Distance __n)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator, typename _Distance >`  
`_RandomAccessIterator __gnu_cxx::__random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out, _RandomNumberGenerator &__rand, const _Distance __n)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`  
`pair< _InputIterator, _OutputIterator > __gnu_cxx::__copy_n (_InputIterator __first, _Size __count, _OutputIterator __result)`
- `template<typename _InputIterator, typename _Tp, typename _Size >`  
`void __gnu_cxx::__count (_InputIterator __first, _InputIterator __last, const _Tp &__value, _Size &__n)`
- `template<typename _InputIterator, typename _Predicate, typename _Size >`  
`void __gnu_cxx::__count_if (_InputIterator __first, _InputIterator __last, _Predicate __pred, _Size &__n)`
- `template<typename _RandomAccessIterator >`  
`bool __gnu_cxx::__is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _StrictWeakOrdering >`  
`bool __gnu_cxx::__is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _StrictWeakOrdering __comp)`
- `template<typename _ForwardIterator, typename _StrictWeakOrdering >`  
`bool __gnu_cxx::__is_sorted (_ForwardIterator __first, _ForwardIterator __last, _StrictWeakOrdering __comp)`
- `template<typename _ForwardIterator >`  
`bool __gnu_cxx::__is_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`int __gnu_cxx::__lexicographical_compare_3way (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator >`  
`_RandomAccessIterator __gnu_cxx::__random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last, _RandomNumberGenerator &__rand)`



- `template<typename _InputIterator, typename _RandomAccessIterator >`  
`_RandomAccessIterator \_\_gnu\_cxx::random\_sample (_InputIterator __`  
`__first, _InputIterator __last, _RandomAccessIterator __out_first, _`  
`RandomAccessIterator __out_last)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance, typename`  
`_RandomNumberGenerator >`  
`_OutputIterator \_\_gnu\_cxx::random\_sample\_n (_ForwardIterator __first, _`  
`ForwardIterator __last, _OutputIterator __out, const _Distance __n, _`  
`RandomNumberGenerator &__rand)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance >`  
`_OutputIterator \_\_gnu\_cxx::random\_sample\_n (_ForwardIterator __first, _`  
`ForwardIterator __last, _OutputIterator __out, const _Distance __n)`

#### 6.4.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [ext/algorithm](#).

## 6.5 algorithm File Reference

#### 6.5.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [parallel/algorithm](#).

## 6.6 algorithmfwd.h File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _Filter >`  
`_Filter std::adjacent\_find (_Filter, _Filter)`
- `template<typename _Filter, typename _BinaryPredicate >`  
`_Filter std::adjacent\_find (_Filter, _Filter, _BinaryPredicate)`
- `template<typename _Iter, typename _Predicate >`  
`bool std::all\_of (_Iter, _Iter, _Predicate)`
- `template<typename _Iter, typename _Predicate >`  
`bool std::any\_of (_Iter, _Iter, _Predicate)`

- `template<typename _Filter, typename _Tp, typename _Compare >`  
`bool std::binary_search (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _Filter, typename _Tp >`  
`bool std::binary_search (_Filter, _Filter, const _Tp &)`
- `template<typename _Iter, typename _OIter >`  
`_OIter std::copy (_Iter, _Iter, _OIter)`
- `template<typename _BIter1, typename _BIter2 >`  
`_BIter2 std::copy_backward (_BIter1, _BIter1, _BIter2)`
- `template<typename _Iter, typename _OIter, typename _Predicate >`  
`_OIter std::copy_if (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _Iter, typename _Size, typename _OIter >`  
`_OIter std::copy_n (_Iter, _Size, _OIter)`
- `template<typename _Iter, typename _Tp >`  
`iterator_traits< _Iter >::difference_type std::count (_Iter, _Iter, const _Tp &)`
  
- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >::difference_type std::count_if (_Iter, _Iter, _-`  
`Predicate)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::equal (_Iter1, _Iter1, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`  
`bool std::equal (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _-`  
`BinaryPredicate __binary_pred)`
- `template<typename _Filter, typename _Tp >`  
`pair< _Filter, _Filter > std::equal_range (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare >`  
`pair< _Filter, _Filter > std::equal_range (_Filter, _Filter, const _Tp &, _-`  
`Compare)`
- `template<typename _Filter, typename _Tp >`  
`void std::fill (_Filter, _Filter, const _Tp &)`
- `template<typename _OIter, typename _Size, typename _Tp >`  
`_OIter std::fill_n (_OIter, _Size, const _Tp &)`
- `template<typename _Iter, typename _Tp >`  
`_Iter std::find (_Iter, _Iter, const _Tp &)`
- `template<typename _Filter1, typename _Filter2 >`  
`_Filter1 std::find_end (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`  
`_Filter1 std::find_end (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _Filter1, typename _Filter2 >`  
`_Filter1 std::find_first_of (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`  
`_Filter1 std::find_first_of (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
  
- `template<typename _Iter, typename _Predicate >`  
`_Iter std::find_if (_Iter, _Iter, _Predicate)`

- `template<typename _Iter, typename _Predicate >`  
`_Iter std::find_if_not (_Iter, _Iter, _Predicate)`
- `template<typename _Iter, typename _Funct >`  
`_Funct std::for_each (_Iter, _Iter, _Funct)`
- `template<typename _FIter, typename _Generator >`  
`void std::generate (_FIter, _FIter, _Generator)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter std::generate_n (_OIter, _Size, _Generator)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::includes (_Iter1, _Iter1, _Iter2, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _Compare >`  
`bool std::includes (_Iter1, _Iter1, _Iter2, _Iter2, _Compare)`
- `template<typename _BIter >`  
`void std::inplace_merge (_BIter, _BIter, _BIter)`
- `template<typename _BIter, typename _Compare >`  
`void std::inplace_merge (_BIter, _BIter, _BIter, _Compare)`
- `template<typename _RAIter >`  
`bool std::is_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`bool std::is_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`  
`_RAIter std::is_heap_until (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`_RAIter std::is_heap_until (_RAIter, _RAIter, _Compare)`
- `template<typename _Iter, typename _Predicate >`  
`bool std::is_partitioned (_Iter, _Iter, _Predicate)`
- `template<typename _FIter1, typename _FIter2 >`  
`bool std::is_permutation (_FIter1, _FIter1, _FIter2)`
- `template<typename _FIter1, typename _FIter2, typename _BinaryPredicate >`  
`bool std::is_permutation (_FIter1, _FIter1, _FIter2, _BinaryPredicate)`
- `template<typename _FIter >`  
`bool std::is_sorted (_FIter, _FIter)`
- `template<typename _FIter, typename _Compare >`  
`bool std::is_sorted (_FIter, _FIter, _Compare)`
- `template<typename _FIter >`  
`_FIter std::is_sorted_until (_FIter, _FIter)`
- `template<typename _FIter, typename _Compare >`  
`_FIter std::is_sorted_until (_FIter, _FIter, _Compare)`
- `template<typename _FIter1, typename _FIter2 >`  
`void std::iter_swap (_FIter1, _FIter2)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::lexicographical_compare (_Iter1, _Iter1, _Iter2, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _Compare >`  
`bool std::lexicographical_compare (_Iter1, _Iter1, _Iter2, _Iter2, _Compare)`

- `template<typename _Filter, typename _Tp >`  
`_Filter std::lower_bound (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare >`  
`_Filter std::lower_bound (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _RAIter >`  
`void std::make_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void std::make_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _Tp >`  
`const _Tp & std::max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`  
`const _Tp & std::max (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`  
`_Tp std::max (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`  
`_Tp std::max (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter >`  
`_Filter std::max_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`  
`_Filter std::max_element (_Filter, _Filter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter std::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Tp >`  
`const _Tp & std::min (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`  
`const _Tp & std::min (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`  
`_Tp std::min (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`  
`_Tp std::min (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter >`  
`_Filter std::min_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`  
`_Filter std::min_element (_Filter, _Filter, _Compare)`
- `template<typename _Tp >`  
`pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`  
`pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`  
`pair< _Tp, _Tp > std::minmax (initializer_list< _Tp >)`

- `template<typename _Tp, typename _Compare >`  
`pair< _Tp, _Tp > std::minmax (initializer_list< _Tp >, _Compare)`
- `template<typename _FIter >`  
`pair< _FIter, _FIter > std::minmax_element (_FIter, _FIter)`
- `template<typename _FIter, typename _Compare >`  
`pair< _FIter, _FIter > std::minmax_element (_FIter, _FIter, _Compare)`
- `template<typename _IIter1, typename _IIter2 >`  
`pair< _IIter1, _IIter2 > std::mismatch (_IIter1, _IIter1, _IIter2)`
- `template<typename _IIter1, typename _IIter2, typename _BinaryPredicate >`  
`pair< _IIter1, _IIter2 > std::mismatch (_IIter1, _IIter1, _IIter2, _-`  
`BinaryPredicate)`
- `template<typename _BIter, typename _Compare >`  
`bool std::next_permutation (_BIter, _BIter, _Compare)`
- `template<typename _BIter >`  
`bool std::next_permutation (_BIter, _BIter)`
- `template<typename _IIter, typename _Predicate >`  
`bool std::none_of (_IIter, _IIter, _Predicate)`
- `template<typename _RAIter >`  
`void std::nth_element (_RAIter, _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void std::nth_element (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`  
`void std::partial_sort (_RAIter, _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void std::partial_sort (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _IIter, typename _RAIter >`  
`_RAIter std::partial_sort_copy (_IIter, _IIter, _RAIter, _RAIter)`
- `template<typename _IIter, typename _RAIter, typename _Compare >`  
`_RAIter std::partial_sort_copy (_IIter, _IIter, _RAIter, _RAIter, _Compare)`
- `template<typename _BIter, typename _Predicate >`  
`_BIter std::partition (_BIter, _BIter, _Predicate)`
- `template<typename _IIter, typename _OIter1, typename _OIter2, typename _Predicate >`  
`pair< _OIter1, _OIter2 > std::partition_copy (_IIter, _IIter, _OIter1, _OIter2,`  
`_Predicate)`
- `template<typename _FIter, typename _Predicate >`  
`_FIter std::partition_point (_FIter, _FIter, _Predicate)`
- `template<typename _RAIter >`  
`void std::pop_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void std::pop_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _BIter >`  
`bool std::prev_permutation (_BIter, _BIter)`
- `template<typename _BIter, typename _Compare >`  
`bool std::prev_permutation (_BIter, _BIter, _Compare)`

- `template<typename _RAIter >`  
`void std::push_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void std::push_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter, typename _Generator >`  
`void std::random_shuffle (_RAIter, _RAIter, _Generator &&)`
- `template<typename _RAIter >`  
`void std::random_shuffle (_RAIter, _RAIter)`
- `template<typename _FIter, typename _Tp >`  
`_FIter std::remove (_FIter, _FIter, const _Tp &)`
- `template<typename _IIter, typename _OIter, typename _Tp >`  
`_OIter std::remove_copy (_IIter, _IIter, _OIter, const _Tp &)`
- `template<typename _IIter, typename _OIter, typename _Predicate >`  
`_OIter std::remove_copy_if (_IIter, _IIter, _OIter, _Predicate)`
- `template<typename _FIter, typename _Predicate >`  
`_FIter std::remove_if (_FIter, _FIter, _Predicate)`
- `template<typename _FIter, typename _Tp >`  
`void std::replace (_FIter, _FIter, const _Tp &, const _Tp &)`
- `template<typename _IIter, typename _OIter, typename _Tp >`  
`_OIter std::replace_copy (_IIter, _IIter, _OIter, const _Tp &, const _Tp &)`
- `template<typename _IIter, typename _OIter, typename _Predicate, typename _Tp >`  
`_OIter std::replace_copy_if (_IIter, _IIter, _OIter, _Predicate, const _Tp &)`
- `template<typename _FIter, typename _Predicate, typename _Tp >`  
`void std::replace_if (_FIter, _FIter, _Predicate, const _Tp &)`
- `template<typename _BIter >`  
`void std::reverse (_BIter, _BIter)`
- `template<typename _BIter, typename _OIter >`  
`_OIter std::reverse_copy (_BIter, _BIter, _OIter)`
- `template<typename _FIter >`  
`void std::rotate (_FIter, _FIter, _FIter)`
- `template<typename _FIter, typename _OIter >`  
`_OIter std::rotate_copy (_FIter, _FIter, _FIter, _OIter)`
- `template<typename _FIter1, typename _FIter2, typename _BinaryPredicate >`  
`_FIter1 std::search (_FIter1, _FIter1, _FIter2, _FIter2, _BinaryPredicate)`
- `template<typename _FIter1, typename _FIter2 >`  
`_FIter1 std::search (_FIter1, _FIter1, _FIter2, _FIter2)`
- `template<typename _FIter, typename _Size, typename _Tp, typename _BinaryPredicate >`  
`_FIter std::search_n (_FIter, _FIter, _Size, const _Tp &, _BinaryPredicate)`
- `template<typename _FIter, typename _Size, typename _Tp >`  
`_FIter std::search_n (_FIter, _FIter, _Size, const _Tp &)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare >`  
`_OIter std::set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter std::set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`

- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter std::set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _`  
`Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter std::set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter,`  
`_Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter std::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _RAIter, typename _UGenerator >`  
`void std::shuffle (_RAIter, _RAIter, _UGenerator &&)`
- `template<typename _RAIter >`  
`void std::sort (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void std::sort (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`  
`void std::sort_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void std::sort_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _BIter, typename _Predicate >`  
`_BIter std::stable_partition (_BIter, _BIter, _Predicate)`
- `template<typename _RAIter, typename _Compare >`  
`void std::stable_sort (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`  
`void std::stable_sort (_RAIter, _RAIter)`
- `template<typename _Tp, size_t _Nm >`  
`void std::swap (_Tp(&)[_Nm], _Tp(&)[_Nm])`
- `template<typename _Tp >`  
`void std::swap (_Tp &__a, _Tp &__b)`
- `template<typename _FIter1, typename _FIter2 >`  
`_FIter2 std::swap_ranges (_FIter1, _FIter1, _FIter2)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BinaryOperation >`  
`_OIter std::transform (_Iter1, _Iter1, _Iter2, _OIter, _BinaryOperation)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`  
`_OIter std::transform (_Iter, _Iter, _OIter, _UnaryOperation)`
- `template<typename _FIter, typename _BinaryPredicate >`  
`_FIter std::unique (_FIter, _FIter, _BinaryPredicate)`
- `template<typename _FIter >`  
`_FIter std::unique (_FIter, _FIter)`

- `template<typename _Iter, typename _OIter, typename _BinaryPredicate >  
_OIter std::unique_copy (_Iter, _Iter, _OIter, _BinaryPredicate)`
- `template<typename _Iter, typename _OIter >  
_OIter std::unique_copy (_Iter, _Iter, _OIter)`
- `template<typename _Filter, typename _Tp, typename _Compare >  
_Filter std::upper_bound (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _Filter, typename _Tp >  
_Filter std::upper_bound (_Filter, _Filter, const _Tp &)`

### 6.6.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<algorithm>`.

Definition in file [bits/algorithmfwd.h](#).

## 6.7 algorithmfwd.h File Reference

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_parallel](#)

### Functions

- `template<typename _Filter, typename _IterTag >  
_Filter std::__parallel::__adjacent_find_switch (_Filter, _Filter, _IterTag)`
- `template<typename _RAIter >  
_RAIter std::__parallel::__adjacent_find_switch (_RAIter __begin, _RAIter  
__end, random_access_iterator_tag)`
- `template<typename _RAIter, typename _BiPredicate >  
_RAIter std::__parallel::__adjacent_find_switch (_RAIter, _RAIter, _-  
BiPredicate, random_access_iterator_tag)`
- `template<typename _Filter, typename _BiPredicate, typename _IterTag >  
_Filter std::__parallel::__adjacent_find_switch (_Filter, _Filter, _BiPredicate,  
_IterTag)`
- `template<typename _Iter, typename _Predicate, typename _IterTag >  
iterator_traits< _Iter >::difference_type std::__parallel::__count_if_switch  
(_Iter, _Iter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Predicate >  
iterator_traits< _RAIter >::difference_type std::__parallel::__count_if -  
switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_  
access_iterator_tag, \_\_gnu\_parallel::\_\_Parallelism __parallelism_tag=\_\_gnu\_  
parallel::parallel\_unbalanced)`



- `template<typename _Iter, typename _Tp, typename _IterTag >`  
`iterator_traits< _Iter >::difference_type std::parallel::__count_switch ( _`  
`_Iter, _Iter, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Tp >`  
`iterator_traits< _RAIter >::difference_type std::parallel::__count_`  
`switch ( _RAIter __begin, _RAIter __end, const _Tp &__value, random_`  
`access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=__gnu_`  
`parallel::parallel_unbalanced)`
- `template<typename _Iter, typename _FIter, typename _IterTag1, typename _IterTag2 >`  
`_Iter std::parallel::__find_first_of_switch ( _Iter, _Iter, _FIter, _FIter, _`  
`IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _FIter, typename _BiPredicate, typename _IterTag >`  
`_RAIter std::parallel::__find_first_of_switch ( _RAIter, _RAIter, _FIter, _`  
`FIter, _BiPredicate, random_access_iterator_tag, _IterTag)`
- `template<typename _Iter, typename _FIter, typename _BiPredicate, typename _IterTag1, type-`  
`name _IterTag2 >`  
`_Iter std::parallel::__find_first_of_switch ( _Iter, _Iter, _FIter, _FIter, _`  
`BiPredicate, _IterTag1, _IterTag2)`
- `template<typename _Iter, typename _Predicate, typename _IterTag >`  
`_Iter std::parallel::__find_if_switch ( _Iter, _Iter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Predicate >`  
`_RAIter std::parallel::__find_if_switch ( _RAIter __begin, _RAIter __end,`  
`_Predicate __pred, random_access_iterator_tag)`
- `template<typename _Iter, typename _Tp, typename _IterTag >`  
`_Iter std::parallel::__find_switch ( _Iter, _Iter, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Tp >`  
`_RAIter std::parallel::__find_switch ( _RAIter __begin, _RAIter __end,`  
`const _Tp &__val, random_access_iterator_tag)`
- `template<typename _Iter, typename _Function, typename _IterTag >`  
`_Function std::parallel::__for_each_switch ( _Iter, _Iter, _Function, _`  
`IterTag)`
- `template<typename _RAIter, typename _Function >`  
`_Function std::parallel::__for_each_switch ( _RAIter __begin, _RAIter _`  
`__end, _Function __f, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism`  
`__parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _OIter, typename _Size, typename _Generator, typename _IterTag >`  
`_OIter std::parallel::__generate_n_switch ( _OIter, _Size, _Generator, _`  
`IterTag)`
- `template<typename _RAIter, typename _Size, typename _Generator >`  
`_RAIter std::parallel::__generate_n_switch ( _RAIter __begin, _Size __n,`  
`_Generator __gen, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism _`  
`__parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _FIter, typename _Generator, typename _IterTag >`  
`void std::parallel::__generate_switch ( _FIter, _FIter, _Generator, _IterTag)`

- `template<typename _RAIter, typename _Generator >`  
`void std::parallel::generate_switch (_RAIter __begin, _RAIter __end, _Generator __gen, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IterTag1, typename _IterTag2 >`  
`bool std::parallel::lexicographical_compare_switch (_Iter1, _Iter1, _Iter2, _Iter2, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`  
`bool std::parallel::lexicographical_compare_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _FIter, typename _Compare, typename _IterTag >`  
`_FIter std::parallel::max_element_switch (_FIter, _FIter, _Compare, _IterTag)`
- `template<typename _RAIter, typename _Compare >`  
`_RAIter std::parallel::max_element_switch (_RAIter __begin, _RAIter __end, _Compare __comp, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare, typename _IterTag1, typename _IterTag2, typename _IterTag3 >`  
`_OIter std::parallel::merge_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter std::parallel::merge_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _FIter, typename _Compare, typename _IterTag >`  
`_FIter std::parallel::min_element_switch (_FIter, _FIter, _Compare, _IterTag)`
- `template<typename _RAIter, typename _Compare >`  
`_RAIter std::parallel::min_element_switch (_RAIter __begin, _RAIter __end, _Compare __comp, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`  
`pair< _RAIter1, _RAIter2 > std::parallel::mismatch_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IterTag1, typename _IterTag2 >`  
`pair< _Iter1, _Iter2 > std::parallel::mismatch_switch (_Iter1, _Iter1, _Iter2, _Iter2, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _FIter, typename _Predicate, typename _IterTag >`  
`_FIter std::parallel::partition_switch (_FIter, _FIter, _Predicate, _IterTag)`

- `template<typename _RAIter, typename _Predicate >`  
`_RAIter std::parallel::partition_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)`
- `template<typename _Filter, typename _Predicate, typename _Tp, typename _IterTag >`  
`void std::parallel::replace_if_switch (_Filter, _Filter, _Predicate, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Predicate, typename _Tp >`  
`void std::parallel::replace_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, const _Tp & __new_value, random_access_iterator_tag, gnu\_parallel::Parallelism __parallelism_tag=gnu\_parallel::parallel\_balanced)`
- `template<typename _Filter, typename _Tp, typename _IterTag >`  
`void std::parallel::replace_switch (_Filter, _Filter, const _Tp &, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Tp >`  
`void std::parallel::replace_switch (_RAIter __begin, _RAIter __end, const _Tp & __old_value, const _Tp & __new_value, random_access_iterator_tag, gnu\_parallel::Parallelism __parallelism_tag=gnu\_parallel::parallel\_balanced)`
- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BiPredicate >`  
`_RAIter std::parallel::search_n_switch (_RAIter, _RAIter, _Integer, const _Tp &, _BiPredicate, random_access_iterator_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate, typename _IterTag >`  
`_Filter std::parallel::search_n_switch (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate, _IterTag)`
- `template<typename _RAIter1, typename _RAIter2 >`  
`_RAIter1 std::parallel::search_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Filter1, typename _Filter2, typename _IterTag1, typename _IterTag2 >`  
`_Filter1 std::parallel::search_switch (_Filter1, _Filter1, _Filter2, _Filter2, _IterTag1, _IterTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _BiPredicate >`  
`_RAIter1 std::parallel::search_switch (_RAIter1, _RAIter1, _RAIter2, _RAIter2, _BiPredicate, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate, typename _IterTag1, typename _IterTag2 >`  
`_Filter1 std::parallel::search_switch (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate, _IterTag1, _IterTag2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3 >`  
`_OIter std::parallel::set_difference_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`

- `template<typename _RAIter1 , typename _RAIter2 , typename _Output_RAIter , typename _Predicate >`  
`_Output_RAIter std::parallel::set_difference_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1 , typename _Iter2 , typename _Predicate , typename _OIter , typename _IterTag1 , typename _IterTag2 , typename _IterTag3 >`  
`_OIter std::parallel::set_intersection_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _RAIter1 , typename _RAIter2 , typename _Output_RAIter , typename _Predicate >`  
`_Output_RAIter std::parallel::set_intersection_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1 , typename _Iter2 , typename _Predicate , typename _OIter , typename _IterTag1 , typename _IterTag2 , typename _IterTag3 >`  
`_OIter std::parallel::set_symmetric_difference_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _RAIter1 , typename _RAIter2 , typename _Output_RAIter , typename _Predicate >`  
`_Output_RAIter std::parallel::set_symmetric_difference_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1 , typename _Iter2 , typename _Predicate , typename _OIter , typename _IterTag1 , typename _IterTag2 , typename _IterTag3 >`  
`_OIter std::parallel::set_union_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _RAIter1 , typename _RAIter2 , typename _Output_RAIter , typename _Predicate >`  
`_Output_RAIter std::parallel::set_union_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter , typename _OIter , typename _UnaryOperation , typename _IterTag1 , typename _IterTag2 >`  
`_OIter std::parallel::transform1_switch (_Iter, _Iter, _OIter, _UnaryOperation, _IterTag1, _IterTag2)`
- `template<typename _RAIter , typename _RAOIter , typename _UnaryOperation >`  
`_RAOIter std::parallel::transform1_switch (_RAIter, _RAIter, _RAOIter, _UnaryOperation, random_access_iterator_tag, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism=__gnu_parallel::parallel_balanced)`

- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BiOperation>`  
`_RAIter3 std::__parallel::__transform2_switch (_RAIter1, _RAIter1, _RAIter2, _RAIter3, _BiOperation, random_access_iterator_tag, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism=\_\_gnu\_parallel::parallel\_balanced)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation, typename _Tag1, typename _Tag2, typename _Tag3>`  
`_OIter std::__parallel::__transform2_switch (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, _Tag1, _Tag2, _Tag3)`
- `template<typename _Iter, typename _OIter, typename _Predicate, typename _IterTag1, typename _IterTag2>`  
`_OIter std::__parallel::__unique_copy_switch (_Iter, _Iter, _OIter, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _RandomAccess_OIter, typename _Predicate>`  
`_RandomAccess_OIter std::__parallel::__unique_copy_switch (_RAIter, _RAIter, _RandomAccess_OIter, _Predicate, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _FIter, typename _BiPredicate>`  
`_FIter std::__parallel::adjacent_find (_FIter, _FIter, _BiPredicate)`
- `template<typename _FIter>`  
`_FIter std::__parallel::adjacent_find (_FIter, _FIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIter, typename _BiPredicate>`  
`_FIter std::__parallel::adjacent_find (_FIter, _FIter, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIter>`  
`_FIter std::__parallel::adjacent_find (_FIter, _FIter)`
- `template<typename _Iter, typename _Tp>`  
`iterator_traits< _Iter >::difference_type std::__parallel::count (_Iter __begin, _Iter __end, const _Tp &__value)`
- `template<typename _Iter, typename _Tp>`  
`iterator_traits< _Iter >::difference_type std::__parallel::count (_Iter __begin, _Iter __end, const _Tp &__value, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp>`  
`iterator_traits< _Iter >::difference_type std::__parallel::count (_Iter __begin, _Iter __end, const _Tp &__value, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Predicate>`  
`iterator_traits< _Iter >::difference_type std::__parallel::count_if (_Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Predicate>`  
`iterator_traits< _Iter >::difference_type std::__parallel::count_if (_Iter __begin, _Iter __end, _Predicate __pred)`

- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >::difference_type std::__parallel::count_if ( _Iter __-`  
`begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::__parallel::equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,`  
`\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool std::__parallel::equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,`  
`_Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::__parallel::equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __-`  
`begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool std::__parallel::equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,`  
`_Predicate __pred)`
- `template<typename _Iter, typename _Tp >`  
`_Iter std::__parallel::find ( _Iter __begin, _Iter __end, const _Tp &__val, -`  
`\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp >`  
`_Iter std::__parallel::find ( _Iter __begin, _Iter __end, const _Tp &__val)`
- `template<typename _Iter, typename _FIter, typename _BiPredicate >`  
`_Iter std::__parallel::find_first_of ( _Iter, _Iter, _FIter, _FIter, _BiPredicate,`  
`\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _FIter, typename _BiPredicate >`  
`_Iter std::__parallel::find_first_of ( _Iter, _Iter, _FIter, _FIter, _BiPredicate)`
- `template<typename _Iter, typename _FIter >`  
`_Iter std::__parallel::find_first_of ( _Iter, _Iter, _FIter, _FIter)`
- `template<typename _Iter, typename _FIter >`  
`_Iter std::__parallel::find_first_of ( _Iter, _Iter, _FIter, _FIter, \_\_gnu-`  
`parallel::sequential\_tag)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter std::__parallel::find_if ( _Iter __begin, _Iter __end, _Predicate __pred,`  
`\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter std::__parallel::find_if ( _Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Function >`  
`_Function std::__parallel::for_each ( _Iter __begin, _Iter __end, _Function`  
`__f, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iterator, typename _Function >`  
`_Function std::__parallel::for_each ( _Iterator __begin, _Iterator __end, -`  
`Function __f, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Function >`  
`_Function std::__parallel::for_each ( _Iter, _Iter, _Function)`
- `template<typename _FIter, typename _Generator >`  
`void std::__parallel::generate ( _FIter, _FIter, _Generator)`

- `template<typename _Filter, typename _Generator >`  
`void std::__parallel::generate (_Filter, _Filter, _Generator, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Generator >`  
`void std::__parallel::generate (_Filter, _Filter, _Generator, \_\_gnu\_parallel::\_\_Parallelism)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter std::__parallel::generate_n (_OIter, _Size, _Generator)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter std::__parallel::generate_n (_OIter, _Size, _Generator, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter std::__parallel::generate_n (_OIter, _Size, _Generator, \_\_gnu\_parallel::\_\_Parallelism)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred)`
- `template<typename _Filter >`  
`_Filter std::__parallel::max_element (_Filter, _Filter, \_\_gnu\_parallel::\_\_Parallelism)`
- `template<typename _Filter >`  
`_Filter std::__parallel::max_element (_Filter, _Filter)`
- `template<typename _Filter >`  
`_Filter std::__parallel::max_element (_Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Compare >`  
`_Filter std::__parallel::max_element (_Filter, _Filter, _Compare)`
- `template<typename _Filter, typename _Compare >`  
`_Filter std::__parallel::max_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::\_\_Parallelism)`
- `template<typename _Filter, typename _Compare >`  
`_Filter std::__parallel::max_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter std::__parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _-`  
`Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::__parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, \_\_gnu\_-`  
`parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter std::__parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _-`  
`Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::__parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _FIter, typename _Compare >`  
`_FIter std::__parallel::min_element (_FIter, _FIter, _Compare, \_\_gnu\_-`  
`parallel::sequential\_tag)`
- `template<typename _FIter >`  
`_FIter std::__parallel::min_element (_FIter, _FIter)`
- `template<typename _FIter >`  
`_FIter std::__parallel::min_element (_FIter, _FIter, \_\_gnu\_-`  
`parallel::sequential\_tag)`
- `template<typename _FIter, typename _Compare >`  
`_FIter std::__parallel::min_element (_FIter, _FIter, _Compare)`
- `template<typename _FIter, typename _Compare >`  
`_FIter std::__parallel::min_element (_FIter, _FIter, _Compare, \_\_gnu\_-`  
`parallel::\_Parallelism)`
- `template<typename _FIter >`  
`_FIter std::__parallel::min_element (_FIter, _FIter, \_\_gnu\_parallel::\_-`  
`Parallelism \_\_parallelism\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1`  
`__end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1`  
`__end1, _Iter2 __begin2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1`  
`__end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1`  
`__end1, _Iter2 __begin2, _Predicate __pred)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter`  
`__end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter`  
`__end)`



- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter`  
`__end, _Compare __comp)`
- `template<typename _RAIter >`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter`  
`__end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _-`  
`_RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _-`  
`_RAIter __end)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _-`  
`_RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _-`  
`_RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Predicate >`  
`_Filter std::__parallel::partition (_Filter, _Filter, _Predicate)`
- `template<typename _Filter, typename _Predicate >`  
`_Filter std::__parallel::partition (_Filter, _Filter, _Predicate, \_\_gnu-`  
`parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, \_\_-`  
`gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, _-`  
`RandomNumberGenerator &&__rand)`
- `template<typename _RAIter >`  
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, _-`  
`RandomNumberGenerator &__rand, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Tp >`  
`void std::__parallel::replace (_Filter, _Filter, const _Tp &, const _Tp &, \_\_-`  
`gnu\_parallel::\_Parallelism)`
- `template<typename _Filter, typename _Tp >`  
`void std::__parallel::replace (_Filter, _Filter, const _Tp &, const _Tp &)`
- `template<typename _Filter, typename _Tp >`  
`void std::__parallel::replace (_Filter, _Filter, const _Tp &, const _Tp &, \_\_-`  
`gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`  
`void std::__parallel::replace_if (_Filter, _Filter, _Predicate, const _Tp &, \_\_-`  
`gnu\_parallel::sequential\_tag)`

- `template<typename _Filter, typename _Predicate, typename _Tp >`  
`void std::__parallel::replace_if (_Filter, _Filter, _Predicate, const _Tp &)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`  
`void std::__parallel::replace_if (_Filter, _Filter, _Predicate, const _Tp &, __gnu_parallel::__Parallelism)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate >`  
`_Filter1 std::__parallel::search (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate >`  
`_Filter1 std::__parallel::search (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate, __gnu_parallel::sequential_tag)`
- `template<typename _Filter1, typename _Filter2 >`  
`_Filter1 std::__parallel::search (_Filter1, _Filter1, _Filter2, _Filter2, __gnu_parallel::sequential_tag)`
- `template<typename _Filter1, typename _Filter2 >`  
`_Filter1 std::__parallel::search (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter, typename _Integer, typename _Tp >`  
`_Filter std::__parallel::search_n (_Filter, _Filter, _Integer, const _Tp &, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate >`  
`_Filter std::__parallel::search_n (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp >`  
`_Filter std::__parallel::search_n (_Filter, _Filter, _Integer, const _Tp &)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate >`  
`_Filter std::__parallel::search_n (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter std::__parallel::set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter std::__parallel::set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter std::__parallel::set_intersection (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_intersection (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)`

- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _-`  
`OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::__parallel::set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _-`  
`OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_symmetric_difference (_Iter1, _Iter1, _Iter2, _-`  
`Iter2, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_symmetric_difference (_Iter1, _Iter1, _Iter2, _-`  
`Iter2, _OIter, _Predicate)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::__parallel::set_symmetric_difference (_Iter1, _Iter1, _Iter2, _-`  
`Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::__parallel::set_symmetric_difference (_Iter1, _Iter1, _Iter2, _-`  
`Iter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _-`  
`Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _-`  
`Predicate)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::__parallel::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::__parallel::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, \_\_-`  
`gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, _Compare __-`  
`comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, _Compare __-`  
`comp)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_-`  
`parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare`  
`__comp, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`  
`_OIter std::__parallel::transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`  
`_OIter std::__parallel::transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`  
`_OIter std::__parallel::transform (_Iter, _Iter, _OIter, _UnaryOperation, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`  
`_OIter std::__parallel::transform (_Iter, _Iter, _OIter, _UnaryOperation, \_\_gnu\_parallel::\_Parallelism)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`  
`_OIter std::__parallel::transform (_Iter, _Iter, _OIter, _UnaryOperation)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`  
`_OIter std::__parallel::transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, \_\_gnu\_parallel::\_Parallelism)`
- `template<typename _Iter, typename _OIter >`  
`_OIter std::__parallel::unique_copy (_Iter, _Iter, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter >`  
`_OIter std::__parallel::unique_copy (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::unique_copy (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _Iter, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::unique_copy (_Iter, _Iter, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`

### 6.7.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [parallel/algorithmfwd.h](#).

## 6.8 allocator.h File Reference

### Classes

- class `std::allocator< _Tp >`  
*The standard allocator, as per [20.4].*  
*Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt04ch11.html>.*
- class `std::allocator< void >`  
*`allocator<void>` specialization.*
- struct `std::allocator_arg_t`  
*[`allocator.tag`]*
- struct `std::uses_allocator< _Tp, _Alloc >`  
*[`allocator.uses.trait`]*

### Namespaces

- namespace `std`

### Functions

- template<typename \_T1, typename \_T2 >  
bool **std::operator!=** (const allocator< \_T1 > &, const allocator< \_T2 > &)
- template<typename \_Tp >  
bool **std::operator!=** (const allocator< \_Tp > &, const allocator< \_Tp > &)
- template<typename \_T1, typename \_T2 >  
bool **std::operator==** (const allocator< \_T1 > &, const allocator< \_T2 > &)
- template<typename \_Tp >  
bool **std::operator==** (const allocator< \_Tp > &, const allocator< \_Tp > &)

### Variables

- constexpr allocator\_arg\_t **std::allocator\_arg**

#### 6.8.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

Definition in file [allocator.h](#).

## 6.9 array File Reference

### Classes

- struct [std::array< \\_Tp, \\_Nm >](#)  
*A standard container for storing a fixed size sequence of elements.*

### Namespaces

- namespace [std](#)

### Functions

- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>  
_Tp & std::get (array< _Tp, _Nm > &__arr)`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>  
const _Tp & std::get (const array< _Tp, _Nm > &__arr)`
- `template<typename _Tp, std::size_t _Nm>  
bool std::operator!= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>  
bool std::operator< (const array< _Tp, _Nm > &__a, const array< _Tp, _Nm > &__b)`
- `template<typename _Tp, std::size_t _Nm>  
bool std::operator<= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>  
bool std::operator== (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>  
bool std::operator> (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>  
bool std::operator>= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>  
void std::swap (array< _Tp, _Nm > &__one, array< _Tp, _Nm > &__two)`

#### 6.9.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [array](#).

## 6.10 array\_allocator.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::array\\_allocator< \\_Tp, \\_Array >](#)  
*An allocator that uses previously allocated memory. This memory can be externally, globally, or otherwise allocated.*
- class [\\_\\_gnu\\_cxx::array\\_allocator\\_base< \\_Tp >](#)  
*Base class.*

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Functions

- [template<typename \\_Tp, typename \\_Array >](#)  
[bool \\_\\_gnu\\_cxx::operator!=](#) (const [array\\_allocator< \\_Tp, \\_Array >](#) &, const [array\\_allocator< \\_Tp, \\_Array >](#) &)
- [template<typename \\_Tp, typename \\_Array >](#)  
[bool \\_\\_gnu\\_cxx::operator==](#) (const [array\\_allocator< \\_Tp, \\_Array >](#) &, const [array\\_allocator< \\_Tp, \\_Array >](#) &)

#### 6.10.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [array\\_allocator.h](#).

## 6.11 assoc\_container.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::basic\\_hash\\_table< Key, Mapped, Hash\\_Fn, Eq\\_Fn, Resize\\_Policy, Store\\_Hash, Tag, Policy\\_TL, Allocator >](#)  
*An abstract basic hash-based associative container.*
- class [\\_\\_gnu\\_pbds::basic\\_tree< Key, Mapped, Tag, Node\\_Update, Policy\\_TL, Allocator >](#)  
*An abstract basic tree-like (tree, trie) associative container.*

- class `__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, Allocator >`  
*A concrete collision-chaining hash-based associative container.*
- class `__gnu_pbds::container_base< Key, Mapped, Tag, Policy_Tl, Allocator >`  
*An abstract basic associative container.*
- class `__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, Allocator >`  
*A concrete general-probing hash-based associative container.*
- class `__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, Allocator >`  
*A list-update based associative container.*
- class `__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, Allocator >`  
*A concrete basic tree-based associative container.*
- class `__gnu_pbds::trie< Key, Mapped, E_Access_Traits, Tag, Node_Update, Allocator >`  
*A concrete basic trie-based associative container.*

## Namespaces

- namespace `__gnu_pbds`

## Defines

- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_CLASS_NAME`
- `#define PB_DS_CLASS_NAME`
- `#define PB_DS_CLASS_NAME`
- `#define PB_DS_TREE_NODE_AND_IT_TRAITS_C_DEC`
- `#define PB_DS_TRIE_NODE_AND_ITS_TRAITS`



### 6.11.1 Detailed Description

Contains associative containers.

Definition in file [assoc\\_container.hpp](#).

## 6.12 atomic File Reference

### Classes

- struct [std::atomic< \\_Tp >](#)  
*atomic /// 29.4.3, Generic atomic type, primary class template.*
- struct [std::atomic< \\_Tp \\* >](#)  
*Partial specialization for pointer types.*
- struct [std::atomic< bool >](#)  
*Explicit specialization for bool.*
- struct [std::atomic< char >](#)  
*Explicit specialization for char.*
- struct [std::atomic< char16\\_t >](#)  
*Explicit specialization for char16\_t.*
- struct [std::atomic< char32\\_t >](#)  
*Explicit specialization for char32\_t.*
- struct [std::atomic< int >](#)  
*Explicit specialization for int.*
- struct [std::atomic< long >](#)  
*Explicit specialization for long.*
- struct [std::atomic< long long >](#)  
*Explicit specialization for long long.*
- struct [std::atomic< short >](#)  
*Explicit specialization for short.*
- struct [std::atomic< signed char >](#)  
*Explicit specialization for signed char.*

- struct `std::atomic< unsigned char >`  
*Explicit specialization for unsigned char.*
- struct `std::atomic< unsigned int >`  
*Explicit specialization for unsigned int.*
- struct `std::atomic< unsigned long >`  
*Explicit specialization for unsigned long.*
- struct `std::atomic< unsigned long long >`  
*Explicit specialization for unsigned long long.*
- struct `std::atomic< unsigned short >`  
*Explicit specialization for unsigned short.*
- struct `std::atomic< wchar_t >`  
*Explicit specialization for wchar\_t.*
- struct `std::atomic_bool`  
*atomic\_bool*

## Namespaces

- namespace `std`

## Functions

- bool `std::atomic_compare_exchange_strong` (atomic\_address \*\_\_a, void \*\*\_\_v1, void \*\_\_v2)
- bool `std::atomic_compare_exchange_strong` (volatile atomic\_address \*\_\_a, void \*\*\_\_v1, void \*\_\_v2)
- bool `std::atomic_compare_exchange_strong` (volatile atomic\_bool \*\_\_a, bool \*\_\_i1, bool \_\_i2)
- bool `std::atomic_compare_exchange_strong` (atomic\_bool \*\_\_a, bool \*\_\_i1, bool \_\_i2)
- template<typename \_ITp >  
bool `std::atomic_compare_exchange_strong` (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \*\_\_i1, \_ITp \_\_i2)
- template<typename \_ITp >  
bool `std::atomic_compare_exchange_strong` (volatile \_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \*\_\_i1, \_ITp \_\_i2)

- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_strong_explicit (__atomic_base< _ITp`  
`> *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2)`
- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_strong_explicit (volatile __atomic_-`  
`base< _ITp > *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_-`  
`order __m2)`
- `bool std::atomic_compare_exchange_strong_explicit (atomic_address *__a,`  
`void **__v1, void *__v2, memory_order __m1, memory_order __m2)`
- `bool std::atomic_compare_exchange_strong_explicit (volatile atomic_-`  
`address *__a, void **__v1, void *__v2, memory_order __m1, memory_order`  
`__m2)`
- `bool std::atomic_compare_exchange_strong_explicit (atomic_bool *__a,`  
`bool *__i1, bool __i2, memory_order __m1, memory_order __m2)`
- `bool std::atomic_compare_exchange_strong_explicit (volatile atomic_bool`  
`*__a, bool *__i1, bool __i2, memory_order __m1, memory_order __m2)`
- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_weak (__atomic_base< _ITp > *__a,`  
`_ITp *__i1, _ITp __i2)`
- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_weak (volatile __atomic_base< _ITp >`  
`*__a, _ITp *__i1, _ITp __i2)`
- `bool std::atomic_compare_exchange_weak (atomic_bool *__a, bool *__i1,`  
`bool __i2)`
- `bool std::atomic_compare_exchange_weak (atomic_address *__a, void **__-`  
`v1, void *__v2)`
- `bool std::atomic_compare_exchange_weak (volatile atomic_bool *__a, bool`  
`*__i1, bool __i2)`
- `bool std::atomic_compare_exchange_weak (volatile atomic_address *__a,`  
`void **__v1, void *__v2)`
- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_weak_explicit (volatile __atomic_-`  
`base< _ITp > *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_-`  
`order __m2)`
- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_weak_explicit (__atomic_base< _ITp`  
`> *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2)`
- `bool std::atomic_compare_exchange_weak_explicit (volatile atomic_address`  
`*__a, void **__v1, void *__v2, memory_order __m1, memory_order __m2)`
- `bool std::atomic_compare_exchange_weak_explicit (atomic_bool *__a, bool`  
`*__i1, bool __i2, memory_order __m1, memory_order __m2)`
- `bool std::atomic_compare_exchange_weak_explicit (atomic_address *__a,`  
`void **__v1, void *__v2, memory_order __m1, memory_order __m2)`

- `bool std::atomic_compare_exchange_weak_explicit` (volatile atomic\_bool \*\_\_a, bool \*\_\_i1, bool \_\_i2, memory\_order \_\_m1, memory\_order \_\_m2)
- `template<typename _ITp >`  
`_ITp std::atomic_exchange` (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i)
- `template<typename _ITp >`  
`_ITp std::atomic_exchange` (volatile \_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i)
- `void * std::atomic_exchange` (atomic\_address \*\_\_a, void \*\_\_v)
- `void * std::atomic_exchange` (volatile atomic\_address \*\_\_a, void \*\_\_v)
- `bool std::atomic_exchange` (atomic\_bool \*\_\_a, bool \_\_i)
- `bool std::atomic_exchange` (volatile atomic\_bool \*\_\_a, bool \_\_i)
- `void * std::atomic_exchange_explicit` (atomic\_address \*\_\_a, void \*\_\_v, memory\_order \_\_m)
- `template<typename _ITp >`  
`_ITp std::atomic_exchange_explicit` (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i, memory\_order \_\_m)
- `template<typename _ITp >`  
`_ITp std::atomic_exchange_explicit` (volatile \_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i, memory\_order \_\_m)
- `void * std::atomic_exchange_explicit` (volatile atomic\_address \*\_\_a, void \*\_\_v, memory\_order \_\_m)
- `bool std::atomic_exchange_explicit` (atomic\_bool \*\_\_a, bool \_\_i, memory\_order \_\_m)
- `bool std::atomic_exchange_explicit` (volatile atomic\_bool \*\_\_a, bool \_\_i, memory\_order \_\_m)
- `void * std::atomic_fetch_add` (atomic\_address \*\_\_a, ptrdiff\_t \_\_d)
- `void * std::atomic_fetch_add` (volatile atomic\_address \*\_\_a, ptrdiff\_t \_\_d)
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_add` (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i)
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_add` (volatile \_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i)
- `void * std::atomic_fetch_add_explicit` (atomic\_address \*\_\_a, ptrdiff\_t \_\_d, memory\_order \_\_m)
- `void * std::atomic_fetch_add_explicit` (volatile atomic\_address \*\_\_a, ptrdiff\_t \_\_d, memory\_order \_\_m)
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_add_explicit` (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i, memory\_order \_\_m)
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_add_explicit` (volatile \_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i, memory\_order \_\_m)
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_and` (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i)

- `template<typename _ITp >`  
`_ITp std::atomic_fetch_and (volatile __atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_and_explicit (__atomic_base< _ITp > *__a, _ITp __i,`  
`memory_order __m)`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_and_explicit (volatile __atomic_base< _ITp > *__a,`  
`_ITp __i, memory_order __m)`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_or (__atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_or (volatile __atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_or_explicit (__atomic_base< _ITp > *__a, _ITp __i,`  
`memory_order __m)`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_or_explicit (volatile __atomic_base< _ITp > *__a, _`  
`_ITp __i, memory_order __m)`
- `void * std::atomic_fetch_sub (atomic_address *__a, ptrdiff_t __d)`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_sub (volatile __atomic_base< _ITp > *__a, _ITp __i)`
- `void * std::atomic_fetch_sub (volatile atomic_address *__a, ptrdiff_t __d)`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_sub (__atomic_base< _ITp > *__a, _ITp __i)`
- `void * std::atomic_fetch_sub_explicit (volatile atomic_address *__a, ptrdiff_t`  
`__d, memory_order __m)`
- `void * std::atomic_fetch_sub_explicit (atomic_address *__a, ptrdiff_t __d,`  
`memory_order __m)`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_sub_explicit (__atomic_base< _ITp > *__a, _ITp __i,`  
`memory_order __m)`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_sub_explicit (volatile __atomic_base< _ITp > *__a,`  
`_ITp __i, memory_order __m)`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_xor (__atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_xor (volatile __atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_xor_explicit (volatile __atomic_base< _ITp > *__a,`  
`_ITp __i, memory_order __m)`

- `template<typename _ITp >`  
`_ITp std::atomic_fetch_xor_explicit` (`__atomic_base< _ITp > *__a`, `_ITp __i`,  
`memory_order __m`)
- `void std::atomic_flag_clear` (`atomic_flag *__a`)
- `void std::atomic_flag_clear` (`volatile atomic_flag *__a`)
- `void std::atomic_flag_clear_explicit` (`volatile atomic_flag *__a`, `memory_order __m`)
- `void std::atomic_flag_clear_explicit` (`atomic_flag *__a`, `memory_order __m`)
- `bool std::atomic_flag_test_and_set` (`atomic_flag *__a`)
- `bool std::atomic_flag_test_and_set` (`volatile atomic_flag *__a`)
- `bool std::atomic_flag_test_and_set_explicit` (`volatile atomic_flag *__a`,  
`memory_order __m`)
- `bool std::atomic_flag_test_and_set_explicit` (`atomic_flag *__a`, `memory_order __m`)
- `template<typename _ITp >`  
`void std::atomic_init` (`volatile __atomic_base< _ITp > *__a`, `_ITp __i`)
- `void std::atomic_init` (`volatile atomic_address *__a`, `void *__v`)
- `template<typename _ITp >`  
`void std::atomic_init` (`__atomic_base< _ITp > *__a`, `_ITp __i`)
- `void std::atomic_init` (`atomic_address *__a`, `void *__v`)
- `void std::atomic_init` (`atomic_bool *__a`, `bool __b`)
- `void std::atomic_init` (`volatile atomic_bool *__a`, `bool __b`)
- `template<typename _ITp >`  
`bool std::atomic_is_lock_free` (`const __atomic_base< _ITp > *__a`)
- `template<typename _ITp >`  
`bool std::atomic_is_lock_free` (`const volatile __atomic_base< _ITp > *__a`)
- `bool std::atomic_is_lock_free` (`const atomic_address *__a`)
- `bool std::atomic_is_lock_free` (`const volatile atomic_bool *__a`)
- `bool std::atomic_is_lock_free` (`const atomic_bool *__a`)
- `bool std::atomic_is_lock_free` (`const volatile atomic_address *__a`)
- `bool std::atomic_load` (`const volatile atomic_bool *__a`)
- `void * std::atomic_load` (`const atomic_address *__a`)
- `template<typename _ITp >`  
`_ITp std::atomic_load` (`const volatile __atomic_base< _ITp > *__a`)
- `template<typename _ITp >`  
`_ITp std::atomic_load` (`const __atomic_base< _ITp > *__a`)
- `void * std::atomic_load` (`const volatile atomic_address *__a`)
- `bool std::atomic_load` (`const atomic_bool *__a`)
- `bool std::atomic_load_explicit` (`const atomic_bool *__a`, `memory_order __m`)
- `void * std::atomic_load_explicit` (`const atomic_address *__a`, `memory_order __m`)
- `template<typename _ITp >`  
`_ITp std::atomic_load_explicit` (`const volatile __atomic_base< _ITp > *__a`,  
`memory_order __m`)

- void \* **std::atomic\_load\_explicit** (const volatile atomic\_address \*\_\_a, memory\_order \_\_m)
- template<typename \_ITp >  
\_ITp **std::atomic\_load\_explicit** (const \_\_atomic\_base< \_ITp > \*\_\_a, memory\_order \_\_m)
- bool **std::atomic\_load\_explicit** (const volatile atomic\_bool \*\_\_a, memory\_order \_\_m)
- void **std::atomic\_store** (atomic\_address \*\_\_a, void \*\_\_v)
- void **std::atomic\_store** (volatile atomic\_address \*\_\_a, void \*\_\_v)
- template<typename \_ITp >  
void **std::atomic\_store** (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i)
- void **std::atomic\_store** (volatile atomic\_bool \*\_\_a, bool \_\_i)
- template<typename \_ITp >  
void **std::atomic\_store** (volatile \_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i)
- void **std::atomic\_store** (atomic\_bool \*\_\_a, bool \_\_i)
- void **std::atomic\_store\_explicit** (atomic\_address \*\_\_a, void \*\_\_v, memory\_order \_\_m)
- template<typename \_ITp >  
void **std::atomic\_store\_explicit** (volatile \_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i, memory\_order \_\_m)
- void **std::atomic\_store\_explicit** (volatile atomic\_address \*\_\_a, void \*\_\_v, memory\_order \_\_m)
- void **std::atomic\_store\_explicit** (atomic\_bool \*\_\_a, bool \_\_i, memory\_order \_\_m)
- template<typename \_ITp >  
void **std::atomic\_store\_explicit** (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i, memory\_order \_\_m)
- void **std::atomic\_store\_explicit** (volatile atomic\_bool \*\_\_a, bool \_\_i, memory\_order \_\_m)

### 6.12.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [atomic](#).

## 6.13 atomic\_0.h File Reference

### Classes

- struct [std::\\_\\_atomic0::\\_\\_atomic\\_base< \\_ITp >](#)  
*Base class for atomic integrals.*

- struct [std::\\_\\_atomic0::atomic\\_address](#)  
*atomic\_address*
- struct [std::\\_\\_atomic0::atomic\\_flag](#)  
*atomic\_flag*

### Namespaces

- namespace [std](#)

### Defines

- `#define _ATOMIC_CMPEXCHNG_(__a, __e, __m, __x)`
- `#define _ATOMIC_LOAD_(__a, __x)`
- `#define _ATOMIC_MEMBER_`
- `#define _ATOMIC_MODIFY_(__a, __o, __m, __x)`
- `#define _ATOMIC_STORE_(__a, __m, __x)`

### Functions

- `__atomic_flag_base * std::__atomic0::__atomic_flag_for_address (const volatile void *__z)`
- `void std::__atomic0::__atomic_flag_wait_explicit (__atomic_flag_base *, memory_order)`
- `_GLIBCXX_BEGIN_EXTERN_C void std::__atomic0::atomic_flag_clear_explicit (__atomic_flag_base *, memory_order)`

#### 6.13.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<atomic>`.

Definition in file [atomic\\_0.h](#).

## 6.14 atomic\_2.h File Reference

### Classes

- struct [std::\\_\\_atomic2::\\_\\_atomic\\_base< \\_ITp >](#)  
*Base class for atomic integrals.*



- struct [std::\\_\\_atomic2::atomic\\_address](#)  
*atomic\_address*
- struct [std::\\_\\_atomic2::atomic\\_flag](#)  
*atomic\_flag*

## Namespaces

- namespace [std](#)

### 6.14.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<atomic>`.

Definition in file [atomic\\_2.h](#).

## 6.15 atomic\_base.h File Reference

### Classes

- struct [std::\\_\\_atomic\\_flag\\_base](#)  
*Base type for atomic\_flag.*

## Namespaces

- namespace [std](#)

### Defines

- `#define _GLIBCXX_ATOMIC_NAMESPACE`
- `#define \_GLIBCXX\_ATOMIC\_PROPERTY`
- `#define ATOMIC_ADDRESS_LOCK_FREE`
- `#define ATOMIC_CHAR16_T_LOCK_FREE`
- `#define ATOMIC_CHAR32_T_LOCK_FREE`
- `#define ATOMIC_CHAR_LOCK_FREE`
- `#define ATOMIC_FLAG_INIT`
- `#define ATOMIC_INT_LOCK_FREE`
- `#define ATOMIC_LLONG_LOCK_FREE`
- `#define ATOMIC_LONG_LOCK_FREE`

- `#define ATOMIC_SHORT_LOCK_FREE`
- `#define ATOMIC_VAR_INIT(_VI)`
- `#define ATOMIC_WCHAR_T_LOCK_FREE`

### Typedefs

- `typedef __atomic_base< char > std::atomic_char`
- `typedef __atomic_base< char16_t > std::atomic_char16_t`
- `typedef __atomic_base< char32_t > std::atomic_char32_t`
- `typedef __atomic_base< int > std::atomic_int`
- `typedef __atomic_base< int_fast16_t > std::atomic_int_fast16_t`
- `typedef __atomic_base< int_fast32_t > std::atomic_int_fast32_t`
- `typedef __atomic_base< int_fast64_t > std::atomic_int_fast64_t`
- `typedef __atomic_base< int_fast8_t > std::atomic_int_fast8_t`
- `typedef __atomic_base< int_least16_t > std::atomic_int_least16_t`
- `typedef __atomic_base< int_least32_t > std::atomic_int_least32_t`
- `typedef __atomic_base< int_least64_t > std::atomic_int_least64_t`
- `typedef __atomic_base< int_least8_t > std::atomic_int_least8_t`
- `typedef __atomic_base< intmax_t > std::atomic_intmax_t`
- `typedef __atomic_base< intptr_t > std::atomic_intptr_t`
- `typedef __atomic_base< long long > std::atomic_llong`
- `typedef __atomic_base< long > std::atomic_long`
- `typedef __atomic_base< ptrdiff_t > std::atomic_ptrdiff_t`
- `typedef __atomic_base< signed char > std::atomic_schar`
- `typedef __atomic_base< short > std::atomic_short`
- `typedef __atomic_base< size_t > std::atomic_size_t`
- `typedef __atomic_base< unsigned char > std::atomic_uchar`
- `typedef __atomic_base< unsigned int > std::atomic_uint`
- `typedef __atomic_base< uint_fast16_t > std::atomic_uint_fast16_t`
- `typedef __atomic_base< uint_fast32_t > std::atomic_uint_fast32_t`
- `typedef __atomic_base< uint_fast64_t > std::atomic_uint_fast64_t`
- `typedef __atomic_base< uint_fast8_t > std::atomic_uint_fast8_t`
- `typedef __atomic_base< uint_least16_t > std::atomic_uint_least16_t`
- `typedef __atomic_base< uint_least32_t > std::atomic_uint_least32_t`
- `typedef __atomic_base< uint_least64_t > std::atomic_uint_least64_t`
- `typedef __atomic_base< uint_least8_t > std::atomic_uint_least8_t`
- `typedef __atomic_base< uintmax_t > std::atomic_uintmax_t`
- `typedef __atomic_base< uintptr_t > std::atomic_uintptr_t`
- `typedef __atomic_base< unsigned long long > std::atomic_ullong`
- `typedef __atomic_base< unsigned long > std::atomic_ulong`
- `typedef __atomic_base< unsigned short > std::atomic_ushort`
- `typedef __atomic_base< wchar_t > std::atomic_wchar_t`
- `typedef enum std::memory_order std::memory_order`

### Enumerations

- enum [std::memory\\_order](#) {  
    memory\_order\_relaxed, memory\_order\_consume, memory\_order\_  
    acquire, memory\_order\_release,  
    memory\_order\_acq\_rel, memory\_order\_seq\_cst }

### Functions

- memory\_order [std::\\_\\_calculate\\_memory\\_order](#) (memory\_order \_\_m)
- template<typename \_Tp >  
\_Tp [std::kill\\_dependency](#) (\_Tp \_\_y)

#### 6.15.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<atomic>`.

Definition in file [atomic\\_base.h](#).

## 6.16 atomic\_word.h File Reference

### Typedefs

- typedef int [\\_Atomic\\_word](#)

#### 6.16.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [atomic\\_word.h](#).

## 6.17 atomicity.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Defines

- #define [\\_GLIBCXX\\_READ\\_MEM\\_BARRIER](#)
- #define [\\_GLIBCXX\\_WRITE\\_MEM\\_BARRIER](#)

## Functions

- static void **\_\_gnu\_cxx::\_\_atomic\_add** (volatile \_Atomic\_word \*\_\_mem, int \_\_val)
- static void **\_\_gnu\_cxx::\_\_atomic\_add\_single** (\_Atomic\_word \*\_\_mem, int \_\_val)
- static \_Atomic\_word **\_\_gnu\_cxx::\_\_attribute\_\_** ((\_\_unused\_\_)) \_\_exchange\_and\_add\_dispatch(\_Atomic\_word \*\_\_mem)
- static \_Atomic\_word **\_\_gnu\_cxx::\_\_exchange\_and\_add** (volatile \_Atomic\_word \*\_\_mem, int \_\_val)
- else return **\_\_gnu\_cxx::\_\_exchange\_and\_add\_single** (\_\_mem, \_\_val)
- static \_Atomic\_word **\_\_gnu\_cxx::\_\_exchange\_and\_add\_single** (\_Atomic\_word \*\_\_mem, int \_\_val)
- static \_Atomic\_word int \_\_val **\_\_gnu\_cxx::if** (\_\_gthread\_active\_p()) return \_\_exchange\_and\_add(\_\_mem

## Variables

- static \_Atomic\_word int \_\_val **\_\_gnu\_cxx::\_\_val**

### 6.17.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [atomicity.h](#).

## 6.18 auto\_ptr.h File Reference

### Classes

- class [std::auto\\_ptr< \\_Tp >](#)  
*A simple smart pointer providing strict ownership semantics.*
- struct [std::auto\\_ptr\\_ref< \\_Tp1 >](#)

### Namespaces

- namespace [std](#)

### 6.18.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

Definition in file [auto\\_ptr.h](#).

## 6.19 backward\_warning.h File Reference

### 6.19.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

Definition in file [backward\\_warning.h](#).

## 6.20 balanced\_quicksort.h File Reference

Implementation of a dynamically load-balanced parallel quicksort.

### Classes

- struct [\\_\\_gnu\\_parallel::\\_\\_QSBThreadLocal<\\_RAIter>](#)  
*Information local to one thread in the parallel quicksort run.*

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- template<typename \_RAIter, typename \_Compare>  
void [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\\_qsb](#) (\_RAIter \_\_begin, \_RAIter \_\_end, \_Compare \_\_comp, \_ThreadIndex \_\_num\_threads)
- template<typename \_RAIter, typename \_Compare>  
void [\\_\\_gnu\\_parallel::\\_\\_qsb\\_conquer](#) (\_QSBThreadLocal<\_RAIter> \*\_\_tls, \_RAIter \_\_begin, \_RAIter \_\_end, \_Compare \_\_comp, \_ThreadIndex \_\_iam, \_ThreadIndex \_\_num\_threads, bool \_\_parent\_wait)
- template<typename \_RAIter, typename \_Compare>  
std::iterator\_traits<\_RAIter>::difference\_type [\\_\\_gnu\\_parallel::\\_\\_qsb\\_divide](#) (\_RAIter \_\_begin, \_RAIter \_\_end, \_Compare \_\_comp, \_ThreadIndex \_\_num\_threads)

- `template<typename _RAIter, typename _Compare >`  
`void __gnu_parallel::__qsb_local_sort_with_helping (_QSBThreadLocal< _-`  
`RAIter > **__tls, _Compare &__comp, _ThreadIndex __iam, bool __wait)`

### 6.20.1 Detailed Description

Implementation of a dynamically load-balanced parallel quicksort. It works in-place and needs only logarithmic extra memory. The algorithm is similar to the one proposed in

P. Tsigas and Y. Zhang. A simple, fast parallel implementation of quicksort and its performance evaluation on SUN enterprise 10000. In 11th Euromicro Conference on Parallel, Distributed and Network-Based Processing, page 372, 2003.

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [balanced\\_quicksort.h](#).

## 6.21 base.h File Reference

Sequential helper functions. This file is a GNU parallel extension to the Standard C++ Library.

### Classes

- class [\\_\\_gnu\\_parallel::\\_\\_binder1st< \\_Operation, \\_FirstArgumentType, \\_-](#)  
[SecondArgumentType, \\_ResultType >](#)  
*Similar to `std::binder1st`, but giving the argument types explicitly.*
- class [\\_\\_gnu\\_parallel::\\_\\_binder2nd< \\_Operation, \\_FirstArgumentType, \\_-](#)  
[SecondArgumentType, \\_ResultType >](#)  
*Similar to `std::binder2nd`, but giving the argument types explicitly.*
- class [\\_\\_gnu\\_parallel::\\_\\_unary\\_negate< \\_Predicate, argument\\_type >](#)  
*Similar to `std::unary_negate`, but giving the argument types explicitly.*
- class [\\_\\_gnu\\_parallel::\\_\\_EqualFromLess< \\_T1, \\_T2, \\_Compare >](#)  
*Constructs predicate for equality from strict weak ordering predicate.*
- struct [\\_\\_gnu\\_parallel::\\_\\_EqualTo< \\_T1, \\_T2 >](#)  
*Similar to `std::equal_to`, but allows two different types.*
- struct [\\_\\_gnu\\_parallel::\\_\\_Less< \\_T1, \\_T2 >](#)  
*Similar to `std::less`, but allows two different types.*

- struct `__gnu_parallel::_Multiplies<_Tp1, _Tp2, _Result >`  
*Similar to `std::multiplies`, but allows two different types.*
- struct `__gnu_parallel::_Plus<_Tp1, _Tp2, _Result >`  
*Similar to `std::plus`, but allows two different types.*
- class `__gnu_parallel::_PseudoSequence<_Tp, _DifferenceTp >`  
*Sequence that conceptually consists of multiple copies of the same element. The copies are not stored explicitly, of course.*
- class `__gnu_parallel::_PseudoSequenceIterator<_Tp, _DifferenceTp >`  
*\_Iterator associated with `__gnu_parallel::_PseudoSequence`. If features the usual random-access iterator functionality.*

## Namespaces

- namespace `__gnu_parallel`
- namespace `__gnu_sequential`
- namespace `std`
- namespace `std::__parallel`

## Defines

- `#define _GLIBCXX_PARALLEL_ASSERT(_Condition)`

## Functions

- void `__gnu_parallel::__decode2` (`_CASable __x`, int &`__a`, int &`__b`)
- `_CASable` `__gnu_parallel::__encode2` (int `__a`, int `__b`)
- `_ThreadIndex` `__gnu_parallel::__get_max_threads` ()
- bool `__gnu_parallel::__is_parallel` (const `_Parallelism __p`)
- template<typename `_RAIter`, typename `_Compare` >  
`_RAIter` `__gnu_parallel::__median_of_three_iterators` (`_RAIter __a`, `_RAIter __b`, `_RAIter __c`, `_Compare __comp`)
- template<typename `_Size` >  
`_Size` `__gnu_parallel::__rd_log2` (`_Size __n`)
- template<typename `_Tp` >  
const `_Tp` & `__gnu_parallel::__max` (const `_Tp` &`__a`, const `_Tp` &`__b`)
- template<typename `_Tp` >  
const `_Tp` & `__gnu_parallel::__min` (const `_Tp` &`__a`, const `_Tp` &`__b`)

### 6.21.1 Detailed Description

Sequential helper functions. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [parallel/base.h](#).

## 6.22 **base.h File Reference**

Sequential helper functions. This file is a GNU profile extension to the Standard C++ Library.

### Namespaces

- namespace [\\_\\_gnu\\_profile](#)
- namespace [std](#)
- namespace [std::\\_\\_profile](#)

### 6.22.1 Detailed Description

Sequential helper functions. This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/base.h](#).

## 6.23 **basic\_file.h File Reference**

### Namespaces

- namespace [std](#)

### 6.23.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ios>`.

Definition in file [basic\\_file.h](#).

## 6.24 **basic\_ios.h File Reference**

### Classes

- class [std::basic\\_ios<\\_CharT, \\_Traits>](#)



*Virtual base class for all stream classes.*

*Most of the member functions called dispatched on stream objects (e.g., `std::cout.foo(bar);`) are consolidated in this class.*

## Namespaces

- namespace [std](#)

## Functions

- `template<typename _Facet >`  
`const _Facet & std::__check_facet (const _Facet * __f)`

### 6.24.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ios>`.

Definition in file [basic\\_ios.h](#).

## 6.25 `basic_ios.tcc` File Reference

## Namespaces

- namespace [std](#)

### 6.25.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ios>`.

Definition in file [basic\\_ios.tcc](#).

## 6.26 `basic_iterator.h` File Reference

Includes the original header files concerned with iterators except for stream iterators. This file is a GNU parallel extension to the Standard C++ Library.

### 6.26.1 Detailed Description

Includes the original header files concerned with iterators except for stream iterators. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [basic\\_iterator.h](#).

## 6.27 `basic_string.h` File Reference

### Classes

- class [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >](#)  
*Managing sequences of characters and character-like objects.*
- struct [std::hash< string >](#)  
*std::hash specialization for string.*
- struct [std::hash< u16string >](#)  
*std::hash specialization for u16string.*
- struct [std::hash< u32string >](#)  
*std::hash specialization for u32string.*
- struct [std::hash< wstring >](#)  
*std::hash specialization for wstring.*

### Namespaces

- namespace [std](#)

### Functions

- [template<typename \\_CharT, typename \\_Traits, typename \\_Alloc >](#)  
[basic\\_istream< \\_CharT, \\_Traits > & std::getline](#) ([basic\\_istream< \\_CharT, \\_Traits > &\\_\\_is](#), [basic\\_string< \\_CharT, \\_Traits, \\_Alloc > &\\_\\_str](#), [\\_CharT \\_\\_delim](#))
- [template<typename \\_CharT, typename \\_Traits, typename \\_Alloc >](#)  
[basic\\_istream< \\_CharT, \\_Traits > & std::getline](#) ([basic\\_istream< \\_CharT, \\_Traits > &\\_\\_is](#), [basic\\_string< \\_CharT, \\_Traits, \\_Alloc > &\\_\\_str](#))
- [template<>](#)  
[basic\\_istream< char > & std::getline](#) ([basic\\_istream< char > &\\_\\_in](#), [basic\\_string< char > &\\_\\_str](#), [char \\_\\_delim](#))

- `template<>`  
`basic_istream< wchar_t > & std::getline (basic_istream< wchar_t > &__in,`  
`basic_string< wchar_t > &__str, wchar_t __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator!= (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`  
`const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator!= (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`  
`const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator!= (const _CharT *__lhs, const basic_string< _CharT, _Traits,`  
`_Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _`  
`CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (basic_string< _`  
`CharT, _Traits, _Alloc > &&__lhs, basic_string< _CharT, _Traits, _Alloc >`  
`&&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (const _CharT *__lhs,`  
`basic_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _`  
`CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc`  
`> &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (const _CharT *__lhs,`  
`const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _`  
`CharT, _Traits, _Alloc > &__lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (_CharT __lhs, basic-`  
`string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (basic_string< _`  
`CharT, _Traits, _Alloc > &&__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (_CharT __lhs, const`  
`basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (basic_string< _`  
`CharT, _Traits, _Alloc > &&__lhs, const basic_string< _CharT, _Traits, _Alloc`  
`> &__rhs)`

- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (basic_string< _`  
`CharT, _Traits, _Alloc > &&__lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string<`  
`_CharT, _Traits, _Alloc > &__lhs, basic_string< _CharT, _Traits, _Alloc >`  
`&&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator< (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`  
`const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator< (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`  
`const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator< (const _CharT *__lhs, const basic_string< _CharT, _Traits,`  
`_Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _`  
`CharT, _Traits > &__os, const basic_string< _CharT, _Traits, _Alloc > &__str)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator<= (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`  
`const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator<= (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`  
`const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator<= (const _CharT *__lhs, const basic_string< _CharT, _`  
`Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`  
`const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`  
`const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, bool >::__type`  
`std::operator== (const basic_string< _CharT > &__lhs, const basic_string<`  
`_CharT > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator== (const _CharT *__lhs, const basic_string< _CharT, _Traits,`  
`_Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator> (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`  
`const basic_string< _CharT, _Traits, _Alloc > &__rhs)`

- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator> (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`  
`const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator> (const _CharT *__lhs, const basic_string< _CharT, _Traits,`  
`_Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator>= (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`  
`const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator>= (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`  
`const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator>= (const _CharT *__lhs, const basic_string< _CharT, _`  
`_Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT,`  
`_Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str)`
- `template<>`  
`basic_istream< char > & std::operator>> (basic_istream< char > &__is,`  
`basic_string< char > &__str)`
- `double std::stod (const string &__str, size_t *__idx=0)`
- `float std::stof (const string &__str, size_t *__idx=0)`
- `int std::stoi (const string &__str, size_t *__idx=0, int __base=10)`
- `long std::stol (const string &__str, size_t *__idx=0, int __base=10)`
- `long double std::stold (const string &__str, size_t *__idx=0)`
- `long long std::stoll (const string &__str, size_t *__idx=0, int __base=10)`
- `unsigned long std::stoul (const string &__str, size_t *__idx=0, int __base=10)`
- `unsigned long long std::stoull (const string &__str, size_t *__idx=0, int __`  
`base=10)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`void std::swap (basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string<`  
`_CharT, _Traits, _Alloc > &__rhs)`
- `string std::to\_string (long __val)`
- `string std::to\_string (int __val)`
- `string std::to\_string (double __val)`
- `string std::to\_string (unsigned long __val)`
- `string std::to\_string (long double __val)`
- `string std::to\_string (unsigned __val)`
- `string std::to\_string (long long __val)`
- `string std::to\_string (unsigned long long __val)`
- `string std::to\_string (float __val)`
- `wstring std::to\_wstring (unsigned long long __val)`
- `wstring std::to\_wstring (int __val)`

- `wstring std::to_wstring` (float \_\_val)
- `wstring std::to_wstring` (long double \_\_val)
- `wstring std::to_wstring` (double \_\_val)
- `wstring std::to_wstring` (unsigned \_\_val)
- `wstring std::to_wstring` (long long \_\_val)
- `wstring std::to_wstring` (unsigned long \_\_val)
- `wstring std::to_wstring` (long \_\_val)

### 6.27.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<string>`.

Definition in file [basic\\_string.h](#).

## 6.28 `basic_string.tcc` File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _CharT, typename _Traits, typename _Alloc >  
basic_istream< _CharT, _Traits > & std::getline` (`basic_istream< _CharT, _Traits > &__is`, `basic_string< _CharT, _Traits, _Alloc > &__str`, `_CharT __delim`)
- `template<typename _CharT, typename _Traits, typename _Alloc >  
basic_string< _CharT, _Traits, _Alloc > std::operator+` (`_CharT __lhs`, `const basic_string< _CharT, _Traits, _Alloc > &__rhs`)
- `template<typename _CharT, typename _Traits, typename _Alloc >  
basic_string< _CharT, _Traits, _Alloc > std::operator+` (`const _CharT *__lhs`, `const basic_string< _CharT, _Traits, _Alloc > &__rhs`)
- `template<typename _CharT, typename _Traits, typename _Alloc >  
basic_istream< _CharT, _Traits > & std::operator>>` (`basic_istream< _CharT, _Traits > &__is`, `basic_string< _CharT, _Traits, _Alloc > &__str`)

### 6.28.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<string>`.

Definition in file [basic\\_string.tcc](#).

## 6.29 `basic_types.hpp` File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::value\\_type\\_base< Key, Mapped, Allocator, false >](#)
- struct [\\_\\_gnu\\_pbds::detail::value\\_type\\_base< Key, Mapped, Allocator, true >](#)
- struct [\\_\\_gnu\\_pbds::detail::value\\_type\\_base< Key, null\\_mapped\\_type, Allocator, false >](#)
- struct [\\_\\_gnu\\_pbds::detail::value\\_type\\_base< Key, null\\_mapped\\_type, Allocator, true >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Defines

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`

#### 6.29.1 Detailed Description

Contains basic types used by containers.

Definition in file [basic\\_types.hpp](#).

## 6.30 `binders.h` File Reference

### Classes

- class [std::binder1st< \\_Operation >](#)  
*One of the [binder functors](#).*
- class [std::binder2nd< \\_Operation >](#)  
*One of the [binder functors](#).*

### Namespaces

- namespace [std](#)

## Functions

- `template<typename _Operation, typename _Tp >`  
`binder1st< _Operation > std::bind1st (const _Operation &__fn, const _Tp &_`  
`_x)`
- `template<typename _Operation, typename _Tp >`  
`binder2nd< _Operation > std::bind2nd (const _Operation &__fn, const _Tp &_`  
`_x)`

### 6.30.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

Definition in file [binders.h](#).

## 6.31 `bitmap_allocator.h` File Reference

### Classes

- class [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_mini\\_vector< \\_Tp >](#)  
*`__mini_vector<>` is a stripped down version of the full-fledged `std::vector<>`.*
- class [\\_\\_gnu\\_cxx::\\_\\_detail::\\_Bitmap\\_counter< \\_Tp >](#)  
*The bitmap counter which acts as the bitmap manipulator, and manages the bit-manipulation functions and the searching and identification functions on the bit-map.*
- class [\\_\\_gnu\\_cxx::\\_\\_detail::\\_Ffit\\_finder< \\_Tp >](#)  
*The class which acts as a predicate for applying the first-fit memory allocation policy for the bitmap allocator.*
- class [\\_\\_gnu\\_cxx::bitmap\\_allocator< \\_Tp >](#)  
*Bitmap Allocator, primary template.*
- class [\\_\\_gnu\\_cxx::free\\_list](#)  
*The free list class for managing chunks of memory to be given to and returned by the [bitmap\\_allocator](#).*

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [\\_\\_gnu\\_cxx::\\_\\_detail](#)



**Defines**

- `#define _BALLOC_ALIGN_BYTES`

**Enumerations**

- `enum { bits_per_byte, bits_per_block }`

**Functions**

- `void __gnu_cxx::__detail::__bit_allocate (size_t *__pmap, size_t __pos) throw ()`
- `void __gnu_cxx::__detail::__bit_free (size_t *__pmap, size_t __pos) throw ()`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >  
_ForwardIterator __gnu_cxx::__detail::__lower_bound (_ForwardIterator __-  
first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _AddrPair >  
size_t __gnu_cxx::__detail::__num_bitmaps (_AddrPair __ap)`
- `template<typename _AddrPair >  
size_t __gnu_cxx::__detail::__num_blocks (_AddrPair __ap)`
- `size_t __gnu_cxx::__Bit_scan_forward (size_t __num)`
- `template<typename _Tp1, typename _Tp2 >  
bool __gnu_cxx::operator!= (const bitmap_allocator< _Tp1 > &, const  
bitmap_allocator< _Tp2 > &) throw ()`
- `template<typename _Tp1, typename _Tp2 >  
bool __gnu_cxx::operator== (const bitmap_allocator< _Tp1 > &, const  
bitmap_allocator< _Tp2 > &) throw ()`

**6.31.1 Detailed Description**

This file is a GNU extension to the Standard C++ Library.

Definition in file `bitmap_allocator.h`.

**6.31.2 Define Documentation****6.31.2.1 `#define _BALLOC_ALIGN_BYTES`**

The constant in the expression below is the alignment required in bytes.

Definition at line 44 of file `bitmap_allocator.h`.

## 6.32 `bitset` File Reference

### Classes

- struct `std::_Base_bitset<_Nw>`
- struct `std::_Base_bitset<0>`
- struct `std::_Base_bitset<1>`
- class `std::bitset<_Nb>`

*The `bitset` class represents a fixed-size sequence of bits.*

- class `std::bitset<_Nb>::reference`
- struct `std::hash<::bitset<_Nb>>`  
*`std::hash` specialization for `bitset`.*

### Namespaces

- namespace `std`

### Defines

- `#define GLIBCXX_BITSET_BITS_PER_WORD`
- `#define GLIBCXX_BITSET_WORDS(__n)`

### Functions

- `template<size_t _Nb>`  
`bitset<_Nb> std::operator& (const bitset<_Nb> &__x, const bitset<_Nb> &__y)`
- `template<size_t _Nb>`  
`bitset<_Nb> std::operator| (const bitset<_Nb> &__x, const bitset<_Nb> &__y)`
- `template<size_t _Nb>`  
`bitset<_Nb> std::operator^ (const bitset<_Nb> &__x, const bitset<_Nb> &__y)`
- `template<class _CharT, class _Traits, size_t _Nb>`  
`std::basic_istream<_CharT, _Traits> & std::operator>> (std::basic_istream<_CharT, _Traits> &__is, bitset<_Nb> &__x)`
- `template<class _CharT, class _Traits, size_t _Nb>`  
`std::basic_ostream<_CharT, _Traits> & std::operator<< (std::basic_ostream<_CharT, _Traits> &__os, const bitset<_Nb> &__x)`

### 6.32.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [bitset](#).

## 6.33 `bitset` File Reference

### Classes

- class [std::\\_\\_debug::bitset<\\_Nb>](#)  
*Class [std::bitset](#) with additional safety/checking/debug instrumentation.*
- struct [std::hash<\\_\\_debug::bitset<\\_Nb>>](#)  
*[std::hash](#) specialization for [bitset](#).*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_debug](#)

### Functions

- `template<size_t _Nb>`  
`bitset<_Nb> std::\_\_debug::operator& (const bitset<_Nb> &__x, const bitset<_Nb> &__y)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`  
`std::basic\_ostream<_CharT, _Traits> & std::\_\_debug::operator<< (std::basic\_ostream<_CharT, _Traits> &__os, const bitset<_Nb> &__x)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`  
`std::basic\_istream<_CharT, _Traits> & std::\_\_debug::operator>> (std::basic\_istream<_CharT, _Traits> &__is, bitset<_Nb> &__x)`
- `template<size_t _Nb>`  
`bitset<_Nb> std::\_\_debug::operator^ (const bitset<_Nb> &__x, const bitset<_Nb> &__y)`
- `template<size_t _Nb>`  
`bitset<_Nb> std::\_\_debug::operator| (const bitset<_Nb> &__x, const bitset<_Nb> &__y)`

### 6.33.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/bitset](#).

## 6.34 bitset File Reference

### Classes

- class [std::\\_\\_profile::bitset< \\_Nb >](#)  
*Class [std::bitset](#) wrapper with performance instrumentation.*
- struct [std::hash< \\_\\_profile::bitset< \\_Nb > >](#)  
*[std::hash](#) specialization for [bitset](#).*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_profile](#)

### Functions

- `template<size_t _Nb>  
bitset< _Nb > std::__profile::operator& (const bitset< _Nb > &__x, const  
bitset< _Nb > &__y)`
- `template<typename _CharT, typename _Traits, size_t _Nb>  
std::basic\_ostream< _CharT, _Traits > & std::__profile::operator<<  
(std::basic\_ostream< _CharT, _Traits > &__os, const bitset< _Nb > &__x)`
- `template<typename _CharT, typename _Traits, size_t _Nb>  
std::basic\_istream< _CharT, _Traits > & std::__profile::operator>>  
(std::basic\_istream< _CharT, _Traits > &__is, bitset< _Nb > &__x)`
- `template<size_t _Nb>  
bitset< _Nb > std::__profile::operator^ (const bitset< _Nb > &__x, const  
bitset< _Nb > &__y)`
- `template<size_t _Nb>  
bitset< _Nb > std::__profile::operator| (const bitset< _Nb > &__x, const  
bitset< _Nb > &__y)`

### 6.34.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/bitset](#).

## 6.35 boost\_concept\_check.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Defines

- `#define _GLIBCXX_CLASS_REQUIRES(_type_var, _ns, _concept)`
- `#define _GLIBCXX_CLASS_REQUIRES2(_type_var1, _type_var2, _ns, _concept)`
- `#define _GLIBCXX_CLASS_REQUIRES3(_type_var1, _type_var2, _type_var3, _ns, _concept)`
- `#define _GLIBCXX_CLASS_REQUIRES4(_type_var1, _type_var2, _type_var3, _type_var4, _ns, _concept)`
- `#define _GLIBCXX_DEFINE_BINARY_OPERATOR_CONSTRAINT(_OP, _NAME)`
- `#define _GLIBCXX_DEFINE_BINARY_PREDICATE_OP_CONSTRAINT(_OP, _NAME)`
- `#define _IsUnused`

### Functions

- `template<class _Tp >`  
`void __gnu_cxx::__aux_require_boolean_expr (const _Tp &__t)`
- `void __gnu_cxx::__error_type_must_be_a_signed_integer_type ()`
- `void __gnu_cxx::__error_type_must_be_an_integer_type ()`
- `void __gnu_cxx::__error_type_must_be_an_unsigned_integer_type ()`
- `template<class _Concept >`  
`void __gnu_cxx::__function_requires ()`

### 6.35.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

Definition in file [boost\\_concept\\_check.h](#).

## 6.36 `c++0x_warning.h` File Reference

### 6.36.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

Definition in file [c++0x\\_warning.h](#).

## 6.37 `c++allocator.h` File Reference

### Defines

- `#define __glibcxx_base_allocator`

### 6.37.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

Definition in file [c++allocator.h](#).

## 6.38 `c++config.h` File Reference

### Namespaces

- namespace [std](#)

### Defines

- `#define __GLIBCXX__`
- `#define __glibcxx_assert(_Condition)`
- `#define __N(msgid)`
- `#define _GLIBCXX_ATOMIC_BUILTINS_1`
- `#define _GLIBCXX_ATOMIC_BUILTINS_2`
- `#define _GLIBCXX_ATOMIC_BUILTINS_4`
- `#define _GLIBCXX_ATOMIC_BUILTINS_8`
- `#define _GLIBCXX_BEGIN_EXTERN_C`
- `#define _GLIBCXX_BEGIN_NAMESPACE_LDBL`
- `#define _GLIBCXX_BEGIN_NAMESPACE_VERSION`
- `#define _GLIBCXX_DEPRECATED`
- `#define _GLIBCXX_END_EXTERN_C`
- `#define _GLIBCXX_END_NAMESPACE_LDBL`

- #define `_GLIBCXX_END_NAMESPACE_VERSION`
- #define `_GLIBCXX_EXTERN_TEMPLATE`
- #define `_GLIBCXX_FAST_MATH`
- #define `_GLIBCXX_HAS_GTHREADS`
- #define `_GLIBCXX_HAVE_ACOSF`
- #define `_GLIBCXX_HAVE_ACOSL`
- #define `_GLIBCXX_HAVE_AS_SYMVER_DIRECTIVE`
- #define `_GLIBCXX_HAVE_ASINF`
- #define `_GLIBCXX_HAVE_ASINL`
- #define `_GLIBCXX_HAVE_ATAN2F`
- #define `_GLIBCXX_HAVE_ATAN2L`
- #define `_GLIBCXX_HAVE_ATANF`
- #define `_GLIBCXX_HAVE_ATANL`
- #define `_GLIBCXX_HAVE_ATTRIBUTE_VISIBILITY`
- #define `_GLIBCXX_HAVE_CEILF`
- #define `_GLIBCXX_HAVE_CEILL`
- #define `_GLIBCXX_HAVE_COMPLEX_H`
- #define `_GLIBCXX_HAVE_COSF`
- #define `_GLIBCXX_HAVE_COSHF`
- #define `_GLIBCXX_HAVE_COSHL`
- #define `_GLIBCXX_HAVE_COSL`
- #define `_GLIBCXX_HAVE_DLFCN_H`
- #define `_GLIBCXX_HAVE_EBADMSG`
- #define `_GLIBCXX_HAVE_ECANCELED`
- #define `_GLIBCXX_HAVE_EIDRM`
- #define `_GLIBCXX_HAVE_ENDIAN_H`
- #define `_GLIBCXX_HAVE_ENODATA`
- #define `_GLIBCXX_HAVE_ENOLINK`
- #define `_GLIBCXX_HAVE_ENOSR`
- #define `_GLIBCXX_HAVE_ENOSTR`
- #define `_GLIBCXX_HAVE_ENOTRECOVERABLE`
- #define `_GLIBCXX_HAVE_ENOTSUP`
- #define `_GLIBCXX_HAVE_EOVERFLOW`
- #define `_GLIBCXX_HAVE_EOWNERDEAD`
- #define `_GLIBCXX_HAVE_EPROTO`
- #define `_GLIBCXX_HAVE_ETIME`
- #define `_GLIBCXX_HAVE_ETXTBSY`
- #define `_GLIBCXX_HAVE_EXECINFO_H`
- #define `_GLIBCXX_HAVE_EXPF`
- #define `_GLIBCXX_HAVE_EXPL`
- #define `_GLIBCXX_HAVE_FABSF`
- #define `_GLIBCXX_HAVE_FABSL`
- #define `_GLIBCXX_HAVE_FENV_H`

- #define `_GLIBCXX_HAVE_FINITE`
- #define `_GLIBCXX_HAVE_FINITEF`
- #define `_GLIBCXX_HAVE_FINITEL`
- #define `_GLIBCXX_HAVE_FLOAT_H`
- #define `_GLIBCXX_HAVE_FLOORF`
- #define `_GLIBCXX_HAVE_FLOORL`
- #define `_GLIBCXX_HAVE_FMODF`
- #define `_GLIBCXX_HAVE_FMODL`
- #define `_GLIBCXX_HAVE_FREXPF`
- #define `_GLIBCXX_HAVE_FREXPL`
- #define `_GLIBCXX_HAVE_GETIPINFO`
- #define `_GLIBCXX_HAVE_GTHR_DEFAULT`
- #define `_GLIBCXX_HAVE_HYPOT`
- #define `_GLIBCXX_HAVE_HYPOTF`
- #define `_GLIBCXX_HAVE_HYPOTL`
- #define `_GLIBCXX_HAVE_ICONv`
- #define `_GLIBCXX_HAVE_INT64_T`
- #define `_GLIBCXX_HAVE_INT64_T_LONG`
- #define `_GLIBCXX_HAVE_INTTYPES_H`
- #define `_GLIBCXX_HAVE_ISINF`
- #define `_GLIBCXX_HAVE_ISINF`
- #define `_GLIBCXX_HAVE_ISINFL`
- #define `_GLIBCXX_HAVE_ISNAN`
- #define `_GLIBCXX_HAVE_ISNANF`
- #define `_GLIBCXX_HAVE_ISNANL`
- #define `_GLIBCXX_HAVE_ISWBLANK`
- #define `_GLIBCXX_HAVE_LC_MESSAGES`
- #define `_GLIBCXX_HAVE_LDEXPF`
- #define `_GLIBCXX_HAVE_LDEXPL`
- #define `_GLIBCXX_HAVE_LIBINTL_H`
- #define `_GLIBCXX_HAVE_LIMIT_AS`
- #define `_GLIBCXX_HAVE_LIMIT_DATA`
- #define `_GLIBCXX_HAVE_LIMIT_FSIZE`
- #define `_GLIBCXX_HAVE_LIMIT_RSS`
- #define `_GLIBCXX_HAVE_LIMIT_VMEM`
- #define `_GLIBCXX_HAVE_LINUX_FUTEX`
- #define `_GLIBCXX_HAVE_LOCALE_H`
- #define `_GLIBCXX_HAVE_LOG10F`
- #define `_GLIBCXX_HAVE_LOG10L`
- #define `_GLIBCXX_HAVE_LOGF`
- #define `_GLIBCXX_HAVE_LOGL`
- #define `_GLIBCXX_HAVE_MBSTATE_T`
- #define `_GLIBCXX_HAVE_MEMORY_H`



- #define **\_GLIBCXX\_HAVE\_MODF**
- #define **\_GLIBCXX\_HAVE\_MODFF**
- #define **\_GLIBCXX\_HAVE\_MODFL**
- #define **\_GLIBCXX\_HAVE\_POLL**
- #define **\_GLIBCXX\_HAVE\_POWF**
- #define **\_GLIBCXX\_HAVE\_POWL**
- #define **\_GLIBCXX\_HAVE\_S\_ISREG**
- #define **\_GLIBCXX\_HAVE\_SETENV**
- #define **\_GLIBCXX\_HAVE\_SINCOS**
- #define **\_GLIBCXX\_HAVE\_SINCOSF**
- #define **\_GLIBCXX\_HAVE\_SINCOSL**
- #define **\_GLIBCXX\_HAVE\_SINF**
- #define **\_GLIBCXX\_HAVE\_SINHF**
- #define **\_GLIBCXX\_HAVE\_SINHL**
- #define **\_GLIBCXX\_HAVE\_SINL**
- #define **\_GLIBCXX\_HAVE\_SQRTF**
- #define **\_GLIBCXX\_HAVE\_SQRTL**
- #define **\_GLIBCXX\_HAVE\_STDBOOL\_H**
- #define **\_GLIBCXX\_HAVE\_STDINT\_H**
- #define **\_GLIBCXX\_HAVE\_STDLIB\_H**
- #define **\_GLIBCXX\_HAVE\_STRERROR\_L**
- #define **\_GLIBCXX\_HAVE\_STRERROR\_R**
- #define **\_GLIBCXX\_HAVE\_STRING\_H**
- #define **\_GLIBCXX\_HAVE\_STRINGS\_H**
- #define **\_GLIBCXX\_HAVE\_STRTOF**
- #define **\_GLIBCXX\_HAVE\_STRTOLD**
- #define **\_GLIBCXX\_HAVE\_STRXFRM\_L**
- #define **\_GLIBCXX\_HAVE\_SYMVER\_SYMBOL\_RENAMING\_RUNTIME\_SUPPORT**
- #define **\_GLIBCXX\_HAVE\_SYS\_IOCTL\_H**
- #define **\_GLIBCXX\_HAVE\_SYS\_IPC\_H**
- #define **\_GLIBCXX\_HAVE\_SYS\_PARAM\_H**
- #define **\_GLIBCXX\_HAVE\_SYS\_RESOURCE\_H**
- #define **\_GLIBCXX\_HAVE\_SYS\_SEM\_H**
- #define **\_GLIBCXX\_HAVE\_SYS\_STAT\_H**
- #define **\_GLIBCXX\_HAVE\_SYS\_TIME\_H**
- #define **\_GLIBCXX\_HAVE\_SYS\_TYPES\_H**
- #define **\_GLIBCXX\_HAVE\_SYS\_UIO\_H**
- #define **\_GLIBCXX\_HAVE\_TANF**
- #define **\_GLIBCXX\_HAVE\_TANHF**
- #define **\_GLIBCXX\_HAVE\_TANHL**
- #define **\_GLIBCXX\_HAVE\_TANL**
- #define **\_GLIBCXX\_HAVE\_TGMATH\_H**

- #define **\_GLIBCXX\_HAVE\_TLS**
- #define **\_GLIBCXX\_HAVE\_UNISTD\_H**
- #define **\_GLIBCXX\_HAVE\_VFWSCANF**
- #define **\_GLIBCXX\_HAVE\_VSWSCANF**
- #define **\_GLIBCXX\_HAVE\_VWSCANF**
- #define **\_GLIBCXX\_HAVE\_WCHAR\_H**
- #define **\_GLIBCXX\_HAVE\_WCSTOF**
- #define **\_GLIBCXX\_HAVE\_WCTYPE\_H**
- #define **\_GLIBCXX\_HAVE\_WRITEV**
- #define **\_GLIBCXX\_HOSTED**
- #define **\_GLIBCXX\_ICONV\_CONST**
- #define **\_GLIBCXX\_INLINE\_VERSION**
- #define **\_GLIBCXX\_NAMESPACE\_LDBL**
- #define **\_GLIBCXX\_PACKAGE\_\_GLIBCXX\_VERSION**
- #define **\_GLIBCXX\_PACKAGE\_BUGREPORT**
- #define **\_GLIBCXX\_PACKAGE\_NAME**
- #define **\_GLIBCXX\_PACKAGE\_STRING**
- #define **\_GLIBCXX\_PACKAGE\_TARNAME**
- #define **\_GLIBCXX\_PACKAGE\_URL**
- #define **\_GLIBCXX\_PSEUDO\_VISIBILITY(V)**
- #define **\_GLIBCXX\_RES\_LIMITS**
- #define **\_GLIBCXX\_STDIO\_EOF**
- #define **\_GLIBCXX\_STDIO\_SEEK\_CUR**
- #define **\_GLIBCXX\_STDIO\_SEEK\_END**
- #define **\_GLIBCXX\_SYMVER**
- #define **\_GLIBCXX\_SYMVER\_GNU**
- #define **\_GLIBCXX\_SYNCHRONIZATION\_HAPPENS\_AFTER(A)**
- #define **\_GLIBCXX\_SYNCHRONIZATION\_HAPPENS\_BEFORE(A)**
- #define **\_GLIBCXX\_USE\_C99**
- #define **\_GLIBCXX\_USE\_C99\_COMPLEX**
- #define **\_GLIBCXX\_USE\_C99\_COMPLEX\_TR1**
- #define **\_GLIBCXX\_USE\_C99\_CTYPE\_TR1**
- #define **\_GLIBCXX\_USE\_C99\_FENV\_TR1**
- #define **\_GLIBCXX\_USE\_C99\_INTTYPES\_TR1**
- #define **\_GLIBCXX\_USE\_C99\_INTTYPES\_WCHAR\_T\_TR1**
- #define **\_GLIBCXX\_USE\_C99\_MATH**
- #define **\_GLIBCXX\_USE\_C99\_MATH\_TR1**
- #define **\_GLIBCXX\_USE\_C99\_STDINT\_TR1**
- #define **\_GLIBCXX\_USE\_DECIMAL\_FLOAT**
- #define **\_GLIBCXX\_USE\_GETTIMEOFDAY**
- #define **\_GLIBCXX\_USE\_LFS**
- #define **\_GLIBCXX\_USE\_LONG\_LONG**
- #define **\_GLIBCXX\_USE\_NLS**

- `#define _GLIBCXX_USE_RANDOM_TR1`
- `#define _GLIBCXX_USE_WCHAR_T`
- `#define _GLIBCXX_VISIBILITY(V)`
- `#define _GLIBCXX_WEAK_DEFINITION`
- `#define LT_OBJDIR`
- `#define STDC_HEADERS`

### Typedefs

- `typedef __PTRDIFF_TYPE__ std::ptrdiff_t`
- `typedef __SIZE_TYPE__ std::size_t`

### Functions

- `typedef std::decltype (nullptr) nullptr_t`

#### 6.38.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

Definition in file [c++config.h](#).

## 6.39 c++io.h File Reference

### Namespaces

- namespace [std](#)

### Typedefs

- `typedef FILE std::__c_file`
- `typedef __gthread_mutex_t std::__c_lock`

#### 6.39.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ios>`.

Definition in file [c++io.h](#).

## 6.40 c++locale.h File Reference

### Namespaces

- namespace [std](#)

### Defines

- `#define _GLIBCXX_C_LOCALE_GNU`
- `#define _GLIBCXX_NUM_CATEGORIES`

### Typedefs

- `typedef __locale_t std::__c_locale`

### Functions

- `int std::__convert_from_v (const __c_locale &__cloc __attribute__((__unused__)), char *__out, const int __size __attribute__((__unused__)), const char *__fmt,...)`

#### 6.40.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

Definition in file [c++locale.h](#).

## 6.41 c++locale\_internal.h File Reference

#### 6.41.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

Definition in file [c++locale\\_internal.h](#).

## 6.42 cassert File Reference

#### 6.42.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `assert.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cassert](#).

## 6.43 `cast.h` File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::\\_\\_Caster<\\_ToType>](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Functions

- `template<typename _ToType, typename _FromType >  
_ToType \_\_gnu\_cxx::\_\_const\_pointer\_cast (const _FromType &__arg)`
- `template<typename _ToType, typename _FromType >  
_ToType \_\_gnu\_cxx::\_\_const\_pointer\_cast (_FromType *__arg)`
- `template<typename _ToType, typename _FromType >  
_ToType \_\_gnu\_cxx::\_\_dynamic\_pointer\_cast (const _FromType &__arg)`
- `template<typename _ToType, typename _FromType >  
_ToType \_\_gnu\_cxx::\_\_dynamic\_pointer\_cast (_FromType *__arg)`
- `template<typename _ToType, typename _FromType >  
_ToType \_\_gnu\_cxx::\_\_reinterpret\_pointer\_cast (const _FromType &__arg)`
- `template<typename _ToType, typename _FromType >  
_ToType \_\_gnu\_cxx::\_\_reinterpret\_pointer\_cast (_FromType *__arg)`
- `template<typename _ToType, typename _FromType >  
_ToType \_\_gnu\_cxx::\_\_static\_pointer\_cast (const _FromType &__arg)`
- `template<typename _ToType, typename _FromType >  
_ToType \_\_gnu\_cxx::\_\_static\_pointer\_cast (_FromType *__arg)`

#### 6.43.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include [<ext/pointer.h>](#).

Definition in file [cast.h](#).

## 6.44 `ccomplex` File Reference

### 6.44.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [ccomplex](#).

## 6.45 `ccomplex` File Reference

### 6.45.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/ccomplex](#).

## 6.46 `cctype` File Reference

### Namespaces

- namespace [std](#)

### 6.46.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `cctype.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cctype](#).

## 6.47 `cctype` File Reference

### 6.47.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cctype](#).

## 6.48 `cerrno` File Reference

### Defines

- `#define` `errno`

### 6.48.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `errno.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cerrno](#).

## 6.49 `cfenv` File Reference

### 6.49.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [cfenv](#).

## 6.50 `cfenv` File Reference

### 6.50.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cfenv](#).

## 6.51 `cfloat` File Reference

### Defines

- `#define` `DECIMAL_DIG`
- `#define` `FLT_EVAL_METHOD`

### 6.51.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `float.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cfloat](#).

## 6.52 cfloat File Reference

### Defines

- `#define` **DECIMAL\_DIG**
- `#define` **FLT\_EVAL\_METHOD**

### 6.52.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cfloat](#).

## 6.53 char\_traits.h File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::\\_Char\\_types<\\_CharT>](#)  
*Mapping from character type to associated types.*
- struct [\\_\\_gnu\\_cxx::char\\_traits<\\_CharT>](#)  
*Base class used to implement [std::char\\_traits](#).*
- struct [std::char\\_traits<\\_CharT>](#)  
*Basis for explicit traits specializations.*
- struct [std::char\\_traits<char>](#)  
*21.1.3.1 [char\\_traits](#) specializations*
- struct [std::char\\_traits<wchar\\_t>](#)  
*21.1.3.2 [char\\_traits](#) specializations*

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)



### 6.53.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<string>`.

Definition in file [char\\_traits.h](#).

## 6.54 checkers.h File Reference

Routines for checking the correctness of algorithm results. This file is a GNU parallel extension to the Standard C++ Library.

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _Iter, typename _Compare >`  
`bool \_\_gnu\_parallel::\_\_is\_sorted (_Iter __begin, _Iter __end, _Compare __-`  
`comp)`

### 6.54.1 Detailed Description

Routines for checking the correctness of algorithm results. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [checkers.h](#).

## 6.55 chrono File Reference

### Classes

- struct [std::chrono::duration< \\_Rep, \\_Period >](#)  
*duration*
- struct [std::chrono::duration\\_values< \\_Rep >](#)  
*duration\_values*
- struct [std::chrono::system\\_clock](#)  
*system\_clock*

- struct `std::chrono::time_point< _Clock, _Dur >`  
*time\_point*
- struct `std::chrono::treat_as_floating_point< _Rep >`  
*treat\_as\_floating\_point*

## Namespaces

- namespace `std`
- namespace `std::chrono`

## Typedefs

- typedef system\_clock **`std::chrono::high_resolution_clock`**
- typedef duration< int, ratio< 3600 > > `std::chrono::hours`
- typedef duration< int64\_t, micro > `std::chrono::microseconds`
- typedef duration< int64\_t, milli > `std::chrono::milliseconds`
- typedef duration< int, ratio< 60 > > `std::chrono::minutes`
- typedef system\_clock **`std::chrono::monotonic_clock`**
- typedef duration< int64\_t, nano > `std::chrono::nanoseconds`
- typedef duration< int64\_t > `std::chrono::seconds`

## Functions

- template<typename \_ToDur, typename \_Rep, typename \_Period >  
constexpr enable\_if< \_\_is\_duration< \_ToDur >::value, \_ToDur >::type  
`std::chrono::duration_cast` (const duration< \_Rep, \_Period > &\_\_d)
- template<typename \_Clock, typename \_Dur1, typename \_Dur2 >  
constexpr bool **`std::chrono::operator!=`** (const time\_point< \_Clock, \_Dur1 >  
&\_\_lhs, const time\_point< \_Clock, \_Dur2 > &\_\_rhs)
- template<typename \_Rep1, typename \_Period1, typename \_Rep2, typename \_Period2 >  
constexpr bool **`std::chrono::operator!=`** (const duration< \_Rep1, \_Period1 >  
&\_\_lhs, const duration< \_Rep2, \_Period2 > &\_\_rhs)
- template<typename \_Rep1, typename \_Period1, typename \_Rep2, typename \_Period2 >  
common\_type< duration< \_Rep1, \_Period1 >, duration< \_Rep2, \_Period2 >  
>::type **`std::chrono::operator%`** (const duration< \_Rep1, \_Period1 > &\_\_lhs,  
const duration< \_Rep2, \_Period2 > &\_\_rhs)
- template<typename \_Rep1, typename \_Period, typename \_Rep2 >  
duration< typename \_\_common\_rep\_type< \_Rep1, typename enable\_  
if<!\_\_is\_duration< \_Rep2 >::value, \_Rep2 >::type >::type, \_Period >  
**`std::chrono::operator%`** (const duration< \_Rep1, \_Period > &\_\_d, const \_  
Rep2 &\_\_s)

- `template<typename _Rep1, typename _Period, typename _Rep2 >`  
`duration< typename __common_rep_type< _Rep1, _Rep2 >::type, _Period >`  
`std::chrono::operator* (const duration< _Rep1, _Period > &__d, const _Rep2`  
`&__s)`
- `template<typename _Rep1, typename _Period, typename _Rep2 >`  
`duration< typename __common_rep_type< _Rep2, _Rep1 >::type, _Period >`  
`std::chrono::operator* (const _Rep1 &__s, const duration< _Rep2, _Period >`  
`&__d)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 >`  
`>::type std::chrono::operator+ (const duration< _Rep1, _Period1 > &__lhs,`  
`const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2 >`  
`time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _-`  
`Period2 > >::type > std::chrono::operator+ (const time_point< _Clock, _-`  
`Dur1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Clock, typename _Dur2 >`  
`time_point< _Clock, typename common_type< duration< _Rep1, _Period1 >, _-`  
`Dur2 > >::type > std::chrono::operator+ (const duration< _Rep1, _Period1 >`  
`&__lhs, const time_point< _Clock, _Dur2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2 >`  
`time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _-`  
`Period2 > >::type > std::chrono::operator- (const time_point< _Clock, _-`  
`Dur1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2 >`  
`common_type< _Dur1, _Dur2 >::type std::chrono::operator- (const time_`  
`point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 >`  
`>::type std::chrono::operator- (const duration< _Rep1, _Period1 > &__lhs,`  
`const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period, typename _Rep2 >`  
`duration< typename __common_rep_type< _Rep1, typename enable_`  
`if<!__is_duration< _Rep2 >::value, _Rep2 >::type >::type, _Period >`  
`std::chrono::operator/ (const duration< _Rep1, _Period > &__d, const _Rep2`  
`&__s)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`common_type< _Rep1, _Rep2 >::type std::chrono::operator/ (const`  
`duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 >`  
`&__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2 >`  
`constexpr bool std::chrono::operator< (const time_point< _Clock, _Dur1 >`  
`&__lhs, const time_point< _Clock, _Dur2 > &__rhs)`

- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`constexpr bool std::chrono::operator< (const duration< _Rep1, _Period1 >`  
`&__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`constexpr bool std::chrono::operator<= (const duration< _Rep1, _Period1 >`  
`&__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2 >`  
`constexpr bool std::chrono::operator<= (const time_point< _Clock, _Dur1 >`  
`&__lhs, const time_point< _Clock, _Dur2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2 >`  
`constexpr bool std::chrono::operator== (const time_point< _Clock, _Dur1 >`  
`&__lhs, const time_point< _Clock, _Dur2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`constexpr bool std::chrono::operator== (const duration< _Rep1, _Period1 >`  
`&__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2 >`  
`constexpr bool std::chrono::operator> (const time_point< _Clock, _Dur1 >`  
`&__lhs, const time_point< _Clock, _Dur2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`constexpr bool std::chrono::operator> (const duration< _Rep1, _Period1 >`  
`&__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2 >`  
`constexpr bool std::chrono::operator>= (const time_point< _Clock, _Dur1 >`  
`&__lhs, const time_point< _Clock, _Dur2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`constexpr bool std::chrono::operator>= (const duration< _Rep1, _Period1 >`  
`&__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _ToDur, typename _Clock, typename _Dur >`  
`constexpr enable_if< __is_duration< _ToDur >::value, time_point< _Clock, _`  
`ToDur > >::type std::chrono::time_point_cast (const time_point< _Clock, _Dur`  
`> &__t)`

### 6.55.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [chrono](#).

## 6.56 cinttypes File Reference

### 6.56.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [cinttypes](#).

## 6.57 `cinttypes` File Reference

### 6.57.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cinttypes](#).

## 6.58 `ciso646` File Reference

### 6.58.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the \*.h implementation files.

This is the C++ version of the Standard C Library header `iso646.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [ciso646](#).

## 6.59 `climits` File Reference

### Defines

- `#define LLONG_MAX`
- `#define LLONG_MIN`
- `#define ULLONG_MAX`

### 6.59.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the \*.h implementation files.

This is the C++ version of the Standard C Library header `limits.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [climits](#).

## 6.60 `climits` File Reference

### Defines

- `#define LLONG_MAX`

- `#define LONGLONG_MIN`
- `#define ULLONG_MAX`

### 6.60.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/climits](#).

## 6.61 `clocale` File Reference

### Namespaces

- namespace [std](#)

### 6.61.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the \*.h implementation files.

This is the C++ version of the Standard C Library header `locale.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [clocale](#).

## 6.62 `cmath` File Reference

### Namespaces

- namespace [std](#)

### Functions

- double **std::abs** (double \_\_x)
- float **std::abs** (float \_\_x)
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
**std::abs** (\_Tp \_\_x)
- long double **std::abs** (long double \_\_x)
- float **std::acos** (float \_\_x)
- long double **std::acos** (long double \_\_x)

- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`std::acos (_Tp __x)`
- `float std::asin (float __x)`
- `long double std::asin (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`std::asin (_Tp __x)`
- `float std::atan (float __x)`
- `long double std::atan (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`std::atan (_Tp __x)`
- `template<typename _Tp, typename _Up >`  
`__gnu_cxx::__promote_2< typename __gnu_cxx::__enable_if< __is_`  
`arithmetic< _Tp >::__value &&__is_arithmetic< _Up >::__value, _Tp`  
`>::__type, _Up >::__type std::atan2 (_Tp __y, _Up __x)`
- `float std::atan2 (float __y, float __x)`
- `long double std::atan2 (long double __y, long double __x)`
- `float std::ceil (float __x)`
- `long double std::ceil (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`std::ceil (_Tp __x)`
- `float std::cos (float __x)`
- `long double std::cos (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`std::cos (_Tp __x)`
- `float std::cosh (float __x)`
- `long double std::cosh (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`std::cosh (_Tp __x)`
- `float std::exp (float __x)`
- `long double std::exp (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`std::exp (_Tp __x)`
- `float std::fabs (float __x)`
- `long double std::fabs (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`std::fabs (_Tp __x)`

- float **std::floor** (float \_\_x)
- long double **std::floor** (long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type  
**std::floor** (\_Tp \_\_x)
- float **std::fmod** (float \_\_x, float \_\_y)
- long double **std::fmod** (long double \_\_x, long double \_\_y)
- float **std::frexp** (float \_\_x, int \*\_\_exp)
- long double **std::frexp** (long double \_\_x, int \*\_\_exp)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type  
**std::frexp** (\_Tp \_\_x, int \*\_\_exp)
- float **std::ldexp** (float \_\_x, int \_\_exp)
- long double **std::ldexp** (long double \_\_x, int \_\_exp)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type  
**std::ldexp** (\_Tp \_\_x, int \_\_exp)
- long double **std::log** (long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type  
**std::log** (\_Tp \_\_x)
- float **std::log** (float \_\_x)
- long double **std::log10** (long double \_\_x)
- float **std::log10** (float \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type  
**std::log10** (\_Tp \_\_x)
- long double **std::modf** (long double \_\_x, long double \*\_\_iptr)
- float **std::modf** (float \_\_x, float \*\_\_iptr)
- template<typename \_Tp, typename \_Up >  
\_\_gnu\_cxx::\_\_promote\_2< typename \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_arithmetic< \_Tp >::\_\_value && \_\_is\_arithmetic< \_Up >::\_\_value, \_Tp >::\_\_type, \_Up >::\_\_type **std::pow** (\_Tp \_\_x, \_Up \_\_y)
- float **std::pow** (float \_\_x, float \_\_y)
- long double **std::pow** (long double \_\_x, long double \_\_y)
- float **std::sin** (float \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type  
**std::sin** (\_Tp \_\_x)
- long double **std::sin** (long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type  
**std::sinh** (\_Tp \_\_x)
- long double **std::sinh** (long double \_\_x)



- float **std::sinh** (float \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type  
**std::sqrt** (\_Tp \_\_x)
- long double **std::sqrt** (long double \_\_x)
- float **std::sqrt** (float \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type  
**std::tan** (\_Tp \_\_x)
- long double **std::tan** (long double \_\_x)
- float **std::tan** (float \_\_x)
- long double **std::tanh** (long double \_\_x)
- float **std::tanh** (float \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type  
**std::tanh** (\_Tp \_\_x)

### 6.62.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the \*.h implementation files.

This is the C++ version of the Standard C Library header `math.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cmath](#).

## 6.63 cmath File Reference

### Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

### Functions

- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type [std::tr1::assoc\\_laguerre](#) (unsigned int \_\_n, unsigned int \_\_m, \_Tp \_\_x)
- float **std::tr1::assoc\_laguerref** (unsigned int \_\_n, unsigned int \_\_m, float \_\_x)
- long double **std::tr1::assoc\_laguerrel** (unsigned int \_\_n, unsigned int \_\_m, long double \_\_x)

- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::assoc\_legendre` (unsigned int \_\_l, unsigned int \_\_m, \_Tp \_\_x)
- `float std::tr1::assoc\_legendref` (unsigned int \_\_l, unsigned int \_\_m, float \_\_x)
- `long double std::tr1::assoc\_legendrel` (unsigned int \_\_l, unsigned int \_\_m, long double \_\_x)
- `template<typename _Tpx, typename _Tpy >`  
`__gnu_cxx::__promote_2< _Tpx, _Tpy >::__type std::tr1::beta` (\_Tpx \_\_x, \_Tpy \_\_y)
- `float std::tr1::betaf` (float \_\_x, float \_\_y)
- `long double std::tr1::betal` (long double \_\_x, long double \_\_y)
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::comp\_ellint\_1` (\_Tp \_\_k)
- `float std::tr1::comp\_ellint\_1f` (float \_\_k)
- `long double std::tr1::comp\_ellint\_1l` (long double \_\_k)
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::comp\_ellint\_2` (\_Tp \_\_k)
- `float std::tr1::comp\_ellint\_2f` (float \_\_k)
- `long double std::tr1::comp\_ellint\_2l` (long double \_\_k)
- `template<typename _Tp, typename _Tpn >`  
`__gnu_cxx::__promote_2< _Tp, _Tpn >::__type std::tr1::comp\_ellint\_3` (\_Tp \_\_k, \_Tpn \_\_nu)
- `float std::tr1::comp\_ellint\_3f` (float \_\_k, float \_\_nu)
- `long double std::tr1::comp\_ellint\_3l` (long double \_\_k, long double \_\_nu)
- `template<typename _Tpa, typename _Tpc, typename _Tp >`  
`__gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type std::tr1::conf\_hyperg` (\_Tpa \_\_a, \_Tpc \_\_c, \_Tp \_\_x)
- `float std::tr1::conf\_hypergf` (float \_\_a, float \_\_c, float \_\_x)
- `long double std::tr1::conf\_hypergl` (long double \_\_a, long double \_\_c, long double \_\_x)
- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl\_bessel\_i` (\_Tpnu \_\_nu, \_Tp \_\_x)
- `float std::tr1::cyl\_bessel\_if` (float \_\_nu, float \_\_x)
- `long double std::tr1::cyl\_bessel\_il` (long double \_\_nu, long double \_\_x)
- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl\_bessel\_j` (\_Tpnu \_\_nu, \_Tp \_\_x)
- `float std::tr1::cyl\_bessel\_jf` (float \_\_nu, float \_\_x)
- `long double std::tr1::cyl\_bessel\_jl` (long double \_\_nu, long double \_\_x)
- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl\_bessel\_k` (\_Tpnu \_\_nu, \_Tp \_\_x)
- `float std::tr1::cyl\_bessel\_kf` (float \_\_nu, float \_\_x)

- long double **std::tr1::cyl\_bessel\_kl** (long double \_\_nu, long double \_\_x)
- template<typename \_Tpnu, typename \_Tp >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tpnu, \_Tp >::\_\_type **std::tr1::cyl\_neumann** (\_Tpnu \_\_nu, \_Tp \_\_x)
- float **std::tr1::cyl\_neumannf** (float \_\_nu, float \_\_x)
- long double **std::tr1::cyl\_neumannl** (long double \_\_nu, long double \_\_x)
- template<typename \_Tp, typename \_Tpp >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Tpp >::\_\_type **std::tr1::ellint\_1** (\_Tp \_\_k, \_Tpp \_\_phi)
- float **std::tr1::ellint\_1f** (float \_\_k, float \_\_phi)
- long double **std::tr1::ellint\_1l** (long double \_\_k, long double \_\_phi)
- template<typename \_Tp, typename \_Tpp >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Tpp >::\_\_type **std::tr1::ellint\_2** (\_Tp \_\_k, \_Tpp \_\_phi)
- float **std::tr1::ellint\_2f** (float \_\_k, float \_\_phi)
- long double **std::tr1::ellint\_2l** (long double \_\_k, long double \_\_phi)
- template<typename \_Tp, typename \_Tpn, typename \_Tpp >  
\_\_gnu\_cxx::\_\_promote\_3< \_Tp, \_Tpn, \_Tpp >::\_\_type **std::tr1::ellint\_3** (\_Tp \_\_k, \_Tpn \_\_nu, \_Tpp \_\_phi)
- float **std::tr1::ellint\_3f** (float \_\_k, float \_\_nu, float \_\_phi)
- long double **std::tr1::ellint\_3l** (long double \_\_k, long double \_\_nu, long double \_\_phi)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::expint** (\_Tp \_\_x)
- float **std::tr1::expintf** (float \_\_x)
- long double **std::tr1::expintl** (long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::hermite** (unsigned int \_\_n, \_Tp \_\_x)
- float **std::tr1::hermitef** (unsigned int \_\_n, float \_\_x)
- long double **std::tr1::hermitel** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tpa, typename \_Tpb, typename \_Tpc, typename \_Tp >  
\_\_gnu\_cxx::\_\_promote\_4< \_Tpa, \_Tpb, \_Tpc, \_Tp >::\_\_type **std::tr1::hyperg** (\_Tpa \_\_a, \_Tpb \_\_b, \_Tpc \_\_c, \_Tp \_\_x)
- float **std::tr1::hypergf** (float \_\_a, float \_\_b, float \_\_c, float \_\_x)
- long double **std::tr1::hypergl** (long double \_\_a, long double \_\_b, long double \_\_c, long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::laguerre** (unsigned int \_\_n, \_Tp \_\_x)
- float **std::tr1::laguerref** (unsigned int \_\_n, float \_\_x)
- long double **std::tr1::laguerrel** (unsigned int \_\_n, long double \_\_x)

- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::legendre` (unsigned int \_\_n, \_Tp \_\_x)
- `float std::tr1::legendref` (unsigned int \_\_n, float \_\_x)
- `long double std::tr1::legendrel` (unsigned int \_\_n, long double \_\_x)
- `float std::tr1::pow` (float \_\_x, float \_\_y)
- `template<typename _Tp, typename _Up >`  
`__gnu_cxx::__promote_2< _Tp, _Up >::__type std::tr1::pow` (\_Tp \_\_x, \_Up \_\_y)
- `double std::tr1::pow` (double \_\_x, double \_\_y)
- `long double std::tr1::pow` (long double \_\_x, long double \_\_y)
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::riemann\_zeta` (\_Tp \_\_x)
- `float std::tr1::riemann\_zetaf` (float \_\_x)
- `long double std::tr1::riemann\_zetal` (long double \_\_x)
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::sph\_bessel` (unsigned int \_\_n, \_Tp \_\_x)
- `float std::tr1::sph\_besself` (unsigned int \_\_n, float \_\_x)
- `long double std::tr1::sph\_bessell` (unsigned int \_\_n, long double \_\_x)
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::sph\_legendre` (unsigned int \_\_l, unsigned int \_\_m, \_Tp \_\_theta)
- `float std::tr1::sph\_legendref` (unsigned int \_\_l, unsigned int \_\_m, float \_\_theta)
- `long double std::tr1::sph\_legendrel` (unsigned int \_\_l, unsigned int \_\_m, long double \_\_theta)
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::sph\_neumann` (unsigned int \_\_n, \_Tp \_\_x)
- `float std::tr1::sph\_neumannf` (unsigned int \_\_n, float \_\_x)
- `long double std::tr1::sph\_neumannl` (unsigned int \_\_n, long double \_\_x)

### 6.63.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cmath](#).

## 6.64 codecvt.h File Reference

### Classes

- `class std::\_\_codecvt\_abstract\_base< _InternT, _ExternT, _StateT >`

*Common base for codecvt functions.*

- class `std::codecvt<_InternT, _ExternT, _StateT>`  
*Primary class template codecvt.*  
*NB: Generic, mostly useless implementation.*
- class `std::codecvt<char, char, mbstate_t>`  
*class `codecvt<char, char, mbstate_t>` specialization.*
- class `std::codecvt<wchar_t, char, mbstate_t>`  
*class `codecvt<wchar_t, char, mbstate_t>` specialization.*
- class `std::codecvt_base`  
*Empty base class for codecvt facet [22.2.1.5].*
- class `std::codecvt_byname<_InternT, _ExternT, _StateT>`  
*class `codecvt_byname` [22.2.1.6].*

## Namespaces

- namespace `std`

### 6.64.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

Definition in file `codecvt.h`.

## 6.65 `codecvt_specializations.h` File Reference

### Classes

- struct `__gnu_cxx::encoding_char_traits<_CharT>`  
*`encoding_char_traits`*
- class `__gnu_cxx::encoding_state`  
*Extension to use `iconv` for dealing with character encodings.*
- class `std::codecvt<_InternT, _ExternT, encoding_state>`  
*`codecvt<InternT, _ExternT, encoding_state>` specialization.*

## Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

## Functions

- `template<typename _Tp >  
size_t std::\_\_iconv\_adaptor (size_t(*__func)(iconv_t, _Tp, size_t *, char **,  
size_t *), iconv_t __cd, char **__inbuf, size_t *__inbytes, char **__outbuf,  
size_t *__outbytes)`

### 6.65.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [codecvt\\_specializations.h](#).

## 6.66 compatibility.h File Reference

### 6.66.1 Detailed Description

This is an internal header file, included by other library sources. You should not attempt to use it directly.

Definition in file [x86\\_64-unknown-linux-gnu/bits/compatibility.h](#).

## 6.67 compatibility.h File Reference

Compatibility layer, mostly concerned with atomic operations. This file is a GNU parallel extension to the Standard C++ Library.

## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

## Functions

- `template<typename _Tp >  
bool \_\_gnu\_parallel::\_\_compare\_and\_swap (volatile _Tp *__ptr, _Tp __-  
comparand, _Tp __replacement)`
- `bool \_\_gnu\_parallel::\_\_compare\_and\_swap\_32 (volatile int32_t *__ptr, int32_t  
__comparand, int32_t __replacement)`

- [bool \\_\\_gnu\\_parallel::\\_\\_compare\\_and\\_swap\\_64](#) (volatile int64\_t \*\_\_ptr, int64\_t \_\_comparand, int64\_t \_\_replacement)
- [template<typename \\_Tp > \\_Tp \\_\\_gnu\\_parallel::\\_\\_fetch\\_and\\_add](#) (volatile \_Tp \*\_\_ptr, \_Tp \_\_addend)
- [int32\\_t \\_\\_gnu\\_parallel::\\_\\_fetch\\_and\\_add\\_32](#) (volatile int32\_t \*\_\_ptr, int32\_t \_\_addend)
- [int64\\_t \\_\\_gnu\\_parallel::\\_\\_fetch\\_and\\_add\\_64](#) (volatile int64\_t \*\_\_ptr, int64\_t \_\_addend)
- [void \\_\\_gnu\\_parallel::\\_\\_yield](#) ()

### 6.67.1 Detailed Description

Compatibility layer, mostly concerned with atomic operations. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [parallel/compatibility.h](#).

## 6.68 compiletime\_settings.h File Reference

Defines on options concerning debugging and performance, at compile-time. This file is a GNU parallel extension to the Standard C++ Library.

### Defines

- [#define \\_GLIBCXX\\_ASSERTIONS](#)
- [#define \\_GLIBCXX\\_CALL\(\\_\\_n\)](#)
- [#define \\_GLIBCXX\\_RANDOM\\_SHUFFLE\\_CONSIDER\\_L1](#)
- [#define \\_GLIBCXX\\_RANDOM\\_SHUFFLE\\_CONSIDER\\_TLB](#)
- [#define \\_GLIBCXX\\_SCALE\\_DOWN\\_FPU](#)
- [#define \\_GLIBCXX\\_VERBOSE\\_LEVEL](#)

### 6.68.1 Detailed Description

Defines on options concerning debugging and performance, at compile-time. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [compiletime\\_settings.h](#).

### 6.68.2 Define Documentation

#### 6.68.2.1 #define \_GLIBCXX\_ASSERTIONS

Switch on many `_GLIBCXX_PARALLEL_ASSERTions` in parallel code. Should be switched on only locally.

Definition at line 61 of file `compiletime_settings.h`.

Referenced by `__gnu_parallel::__qsb_local_sort_with_helping()`.

### 6.68.2.2 `#define _GLIBCXX_CALL( __n )`

Macro to produce log message when entering a function.

#### Parameters

`__n` Input size.

#### See also

[\\_GLIBCXX\\_VERBOSE\\_LEVEL](#)

Definition at line 44 of file `compiletime_settings.h`.

Referenced by `__gnu_parallel::__find_template()`, `__gnu_parallel::__for_each_template_random_access_workstealing()`, `__gnu_parallel::__merge_advance()`, `__gnu_parallel::__parallel_nth_element()`, `__gnu_parallel::__parallel_partial_sum()`, `__gnu_parallel::__parallel_partition()`, `__gnu_parallel::__parallel_random_shuffle_drs()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort_qs()`, `__gnu_parallel::__parallel_sort_qsb()`, `__gnu_parallel::__parallel_unique_copy()`, `__gnu_parallel::__search_template()`, `__gnu_parallel::__sequential_multiway_merge()`, `__gnu_parallel::__multiseq_partition()`, `__gnu_parallel::__multiseq_selection()`, `__gnu_parallel::__multiway_merge()`, `__gnu_parallel::__multiway_merge_3_variant()`, `__gnu_parallel::__multiway_merge_4_variant()`, `__gnu_parallel::__multiway_merge_loser_tree()`, `__gnu_parallel::__multiway_merge_loser_tree_sentinel()`, `__gnu_parallel::__multiway_merge_loser_tree_unguarded()`, `__gnu_parallel::__multiway_merge_sentinels()`, `__gnu_parallel::__parallel_multiway_merge()`, and `__gnu_parallel::__parallel_sort_mwms()`.

### 6.68.2.3 `#define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1`

Switch on many `_GLIBCXX_PARALLEL_ASSERTions` in parallel code. Consider the size of the L1 cache for `__gnu_parallel::__parallel_random_shuffle()`.

Definition at line 68 of file `compiletime_settings.h`.



#### 6.68.2.4 `#define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB`

Switch on many `_GLIBCXX_PARALLEL_ASSERTions` in parallel code. Consider the size of the TLB for `gnu_parallel::__parallel_random_shuffle()`.

Definition at line 74 of file `comPILEtime_settings.h`.

#### 6.68.2.5 `#define _GLIBCXX_SCALE_DOWN_FPU`

Use floating-point scaling instead of modulo for mapping random numbers to a range. This can be faster on certain CPUs.

Definition at line 55 of file `comPILEtime_settings.h`.

#### 6.68.2.6 `#define _GLIBCXX_VERBOSE_LEVEL`

Determine verbosity level of the parallel mode. Level 1 prints a message each time a parallel-mode function is entered.

Definition at line 37 of file `comPILEtime_settings.h`.

## 6.69 complex File Reference

### Classes

- struct [std::complex<\\_Tp>](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

### Functions

- [template<typename \\_Tp>  
\\_Tp std::\\_\\_complex\\_abs](#) (const [complex<\\_Tp>](#) &\_\_z)
- [template<typename \\_Tp>  
std::complex<\\_Tp> std::\\_\\_complex\\_acos](#) (const [std::complex<\\_Tp>](#) &\_\_z)

- `template<typename _Tp >`  
`std::complex< _Tp > std::__complex_acosh (const std::complex< _Tp > &_`  
`_z)`
- `template<typename _Tp >`  
`_Tp std::__complex_arg (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::__complex_asin (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::__complex_asinh (const std::complex< _Tp > &_`  
`_z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::__complex_atan (const std::complex< _Tp > &__`  
`z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::__complex_atanh (const std::complex< _Tp > &_`  
`_z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_cos (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_cosh (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_exp (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_log (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_pow (const complex< _Tp > &__x, const`  
`complex< _Tp > &__y)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::__complex_proj (const std::complex< _Tp > &__`  
`z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_sin (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_sinh (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_sqrt (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_tan (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_tanh (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`_Tp std::abs (const complex< _Tp > &)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::acos (const std::complex< _Tp > &__z)`

- `template<typename _Tp >`  
`std::complex< _Tp > std::acosh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`_Tp std::arg (const complex< _Tp > &)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::arg (_Tp __x)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::asin (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::asinh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::atan (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::atanh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::conj (const complex< _Tp > &)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::conj (_Tp __x)`
- `template<typename _Tp >`  
`complex< _Tp > std::cos (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::cosh (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::exp (const complex< _Tp > &)`
- `template<typename _Tp >`  
`_Tp std::fabs (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`constexpr _Tp std::imag (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::imag (_Tp)`
- `template<typename _Tp >`  
`complex< _Tp > std::log (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::log10 (const complex< _Tp > &)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::norm (_Tp __x)`
- `template<typename _Tp >`  
`_Tp std::norm (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator+ (const complex< _Tp > &__x)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator- (const complex< _Tp > &__x)`
- `template<typename _Tp, typename _CharT, class _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-  
CharT, _Traits > &__os, const complex< _Tp > &__x)`

- `template<typename _Tp, typename _CharT, class _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT,`  
`_Traits > &__is, complex< _Tp > &__x)`
- `template<typename _Tp >`  
`complex< _Tp > std::polar (const _Tp &, const _Tp &=0)`
- `template<typename _Tp, typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >`  
`std::pow (const std::complex< _Tp > &__x, const _Up &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::pow (const complex< _Tp > &, const _Tp &)`
- `template<typename _Tp, typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >`  
`std::pow (const _Tp &__x, const std::complex< _Up > &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::pow (const _Tp &, const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::pow (const complex< _Tp > &, const complex< _Tp >`  
`&)`
- `template<typename _Tp, typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >`  
`std::pow (const std::complex< _Tp > &__x, const std::complex< _Up > &__`  
`__y)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::proj (_Tp __x)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::proj (const std::complex< _Tp > &)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::real (_Tp __x)`
- `template<typename _Tp >`  
`constexpr _Tp std::real (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::sin (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::sinh (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::sqrt (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::tan (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::tanh (const complex< _Tp > &)`
  
- `template<typename _Tp >`  
`complex< _Tp > std::operator+ (const complex< _Tp > &__x, const`  
`complex< _Tp > &__y)`

- `template<typename _Tp >`  
`complex< _Tp > std::operator+ (const complex< _Tp > &__x, const _Tp`  
`&__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator+ (const _Tp &__x, const complex< _Tp >`  
`&__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator- (const complex< _Tp > &__x, const`  
`complex< _Tp > &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator- (const complex< _Tp > &__x, const _Tp`  
`&__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator- (const _Tp &__x, const complex< _Tp >`  
`&__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator* (const complex< _Tp > &__x, const`  
`complex< _Tp > &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator* (const complex< _Tp > &__x, const _Tp`  
`&__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator* (const _Tp &__x, const complex< _Tp >`  
`&__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator/ (const complex< _Tp > &__x, const`  
`complex< _Tp > &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator/ (const complex< _Tp > &__x, const _Tp`  
`&__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator/ (const _Tp &__x, const complex< _Tp >`  
`&__y)`
- `template<typename _Tp >`  
`constexpr bool std::operator== (const complex< _Tp > &__x, const`  
`complex< _Tp > &__y)`
- `template<typename _Tp >`  
`constexpr bool std::operator== (const complex< _Tp > &__x, const _Tp &__`  
`__y)`
- `template<typename _Tp >`  
`constexpr bool std::operator== (const _Tp &__x, const complex< _Tp > &__`  
`__y)`

- `template<typename _Tp >`  
`constexpr bool std::operator!= (const complex< _Tp > &__x, const`  
`complex< _Tp > &__y)`
- `template<typename _Tp >`  
`constexpr bool std::operator!= (const complex< _Tp > &__x, const _Tp &__-`  
`__y)`
- `template<typename _Tp >`  
`constexpr bool std::operator!= (const _Tp &__x, const complex< _Tp > &__-`  
`__y)`

### 6.69.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [complex](#).

## 6.70 complex File Reference

### Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

### Functions

- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::\_\_complex\_acos (const std::complex< _Tp >`  
`&__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::\_\_complex\_acosh (const std::complex< _Tp >`  
`&__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::\_\_complex\_asin (const std::complex< _Tp >`  
`&__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::\_\_complex\_asinh (const std::complex< _Tp >`  
`&__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::\_\_complex\_atan (const std::complex< _Tp >`  
`&__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::\_\_complex\_atanh (const std::complex< _Tp >`  
`&__z)`

- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::acos (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::acosh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::asin (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::asinh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::atan (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::atanh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::conj (const complex< _Tp > &)`
- `template<typename _Tp >`  
`std::complex< typename __gnu_cxx::__promote< _Tp >::__type >`  
`std::tr1::conj (_Tp __x)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::fabs (const std::complex< _Tp > &__z)`
- `template<typename _Tp, typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >`  
`std::tr1::polar (const _Tp &__rho, const _Up &__theta)`
- `template<typename _Tp >`  
`complex< _Tp > std::pow (const complex< _Tp > &, const _Tp &)`
- `template<typename _Tp, typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >`  
`std::tr1::pow (const std::complex< _Tp > &__x, const _Up &__y)`
- `template<typename _Tp, typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >`  
`std::tr1::pow (const _Tp &__x, const std::complex< _Up > &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::pow (const complex< _Tp > &, const complex< _Tp > &)`
- `template<typename _Tp, typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >`  
`std::tr1::pow (const std::complex< _Tp > &__x, const std::complex< _Up > &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::pow (const _Tp &, const complex< _Tp > &)`

### 6.70.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/complex](#).

## 6.71 `complex.h` File Reference

### 6.71.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [complex.h](#).

## 6.72 `concept_check.h` File Reference

### Defines

- `#define __glibcxx_class_requires(_a, _b)`
- `#define __glibcxx_class_requires2(_a, _b, _c)`
- `#define __glibcxx_class_requires3(_a, _b, _c, _d)`
- `#define __glibcxx_class_requires4(_a, _b, _c, _d, _e)`
- `#define __glibcxx_function_requires(...)`

### 6.72.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

Definition in file [concept\\_check.h](#).

## 6.73 `concurrency.h` File Reference

### Classes

- class [\\_\\_gnu\\_cxx::\\_\\_scoped\\_lock](#)  
*Scoped lock idiom.*

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Enumerations

- enum `_Lock_policy` { `_S_single`, `_S_mutex`, `_S_atomic` }



### Functions

- void `__gnu_cxx::__throw_concurrency_lock_error()`
- void `__gnu_cxx::__throw_concurrency_unlock_error()`

### Variables

- static const `_Lock_policy` `__gnu_cxx::__default_lock_policy`

#### 6.73.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [concurrency.h](#).

## 6.74 cond\_dealtor.hpp File Reference

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Defines

- `#define PB_DS_COND DEALTOR_CLASS_C_DEC`
- `#define PB_DS_COND DEALTOR_CLASS_T_DEC`

#### 6.74.1 Detailed Description

Contains a conditional deallocator.

Definition in file [cond\\_dealtor.hpp](#).

## 6.75 condition\_variable File Reference

### Classes

- class [std::condition\\_variable](#)  
*condition\_variable*
- class [std::condition\\_variable\\_any](#)  
*condition\_variable\_any*

## Namespaces

- namespace [std](#)

## Enumerations

- enum [std::cv\\_status](#) { [no\\_timeout](#), [timeout](#) }

### 6.75.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [condition\\_variable](#).

## 6.76 constructors\_destructor\_fn\_imps.hpp File Reference

### Functions

- **PB\_DS\_CLASS\_NAME** ()
- `template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 ,  
typename T6 , typename T7 , typename T8 >  
PB_DS_CLASS_NAME (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6, T7 t7,  
T8 t8)`
- `template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 ,  
typename T6 , typename T7 >  
PB_DS_CLASS_NAME (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6, T7 t7)`
- `template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 ,  
typename T6 >  
PB_DS_CLASS_NAME (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6)`
- `template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5  
>  
PB_DS_CLASS_NAME (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5)`
- `template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 >  
PB_DS_CLASS_NAME (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4)`
- `template<typename T0 , typename T1 , typename T2 , typename T3 >  
PB_DS_CLASS_NAME (T0 t0, T1 t1, T2 t2, T3 t3)`
- `template<typename T0 , typename T1 , typename T2 >  
PB_DS_CLASS_NAME (T0 t0, T1 t1, T2 t2)`
- `template<typename T0 , typename T1 >  
PB_DS_CLASS_NAME (T0 t0, T1 t1)`
- `template<typename T0 >  
PB_DS_CLASS_NAME (T0 t0)`
- **PB\_DS\_CLASS\_NAME** (const PB\_DS\_CLASS\_NAME &other)

### 6.76.1 Detailed Description

Contains constructors\_destructor\_fn\_imps applicable to different containers.

Definition in file [constructors\\_destructor\\_fn\\_imps.hpp](#).

## 6.77 container\_base\_dispatch.hpp File Reference

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 6.77.1 Detailed Description

Contains an associative container dispatching base.

Definition in file [container\\_base\\_dispatch.hpp](#).

## 6.78 cpp\_type\_traits.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

### 6.78.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/type_traits>`.

Definition in file [cpp\\_type\\_traits.h](#).

## 6.79 cpu\_defines.h File Reference

### 6.79.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

Definition in file [cpu\\_defines.h](#).

## 6.80 `csetjmp` File Reference

### Namespaces

- namespace [std](#)

### Defines

- `#define setjmp(env)`

#### 6.80.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `setjmp.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [csetjmp](#).

## 6.81 `csignal` File Reference

### Namespaces

- namespace [std](#)

#### 6.81.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `signal.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [csignal](#).

## 6.82 `cstdarg` File Reference

### Namespaces

- namespace [std](#)

**Defines**

- `#define va_end(ap)`

**6.82.1 Detailed Description**

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `stdarg.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cstdarg](#).

**6.83 cstdarg File Reference****6.83.1 Detailed Description**

This is a TR1 C++ Library header.

Definition in file [tr1/cstdarg](#).

**6.84 cstdbool File Reference****6.84.1 Detailed Description**

This is a Standard C++ Library header.

Definition in file [cstdbool](#).

**6.85 cstdbool File Reference****6.85.1 Detailed Description**

This is a TR1 C++ Library header.

Definition in file [tr1/cstdbool](#).

## 6.86 `cstdint` File Reference

### 6.86.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `stdint.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cstdint](#).

## 6.87 `cstdint` File Reference

### Namespaces

- namespace [std](#)

### 6.87.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [cstdint](#).

## 6.88 `cstdint` File Reference

### Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

### 6.88.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cstdint](#).

## 6.89 `cstdio` File Reference

### Namespaces

- namespace [std](#)

### 6.89.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `stdio.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cstdio](#).

## 6.90 `cstdio` File Reference

### 6.90.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cstdio](#).

## 6.91 `cstdlib` File Reference

### Namespaces

- namespace [std](#)

### Defines

- `#define EXIT_FAILURE`
- `#define EXIT_SUCCESS`

### Functions

- `void std::abort` (void) throw ()
- `int std::atexit` (void(\*)()) throw ()
- `void std::exit` (int) throw ()

### 6.91.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `stdlib.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cstdlib](#).

## 6.92 `cstdlib` File Reference

### 6.92.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cstdlib](#).

## 6.93 `cstring` File Reference

### Namespaces

- namespace [std](#)

### Functions

- void \* **std::memchr** (void \*\_\_s, int \_\_c, size\_t \_\_n)
- char \* **std::strchr** (char \*\_\_s, int \_\_n)
- char \* **std::strpbrk** (char \*\_\_s1, const char \*\_\_s2)
- char \* **std::strrchr** (char \*\_\_s, int \_\_n)
- char \* **std::strstr** (char \*\_\_s1, const char \*\_\_s2)

### 6.93.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the \*.h implementation files.

This is the C++ version of the Standard C Library header `string.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cstring](#).

## 6.94 `ctgmath` File Reference

### 6.94.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [ctgmath](#).



## 6.95 `ctgmath` File Reference

### 6.95.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/ctgmath](#).

## 6.96 `ctime` File Reference

### Namespaces

- namespace [std](#)

### 6.96.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `time.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [ctime](#).

## 6.97 `ctime` File Reference

### 6.97.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/ctime](#).

## 6.98 `ctype_base.h` File Reference

### Classes

- struct [std::ctype\\_base](#)  
*Base class for `ctype`.*

### Namespaces

- namespace [std](#)

### 6.98.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

Definition in file [ctype\\_base.h](#).

## 6.99 ctype\_inline.h File Reference

### Namespaces

- namespace [std](#)

### 6.99.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

Definition in file [ctype\\_inline.h](#).

## 6.100 ctype\_noninline.h File Reference

### 6.100.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

Definition in file [ctype\\_noninline.h](#).

## 6.101 cwchar File Reference

### Namespaces

- namespace [std](#)

### Functions

- `wchar_t * std::wcschr (wchar_t *__p, wchar_t __c)`
- `wchar_t * std::wcpbrk (wchar_t *__s1, const wchar_t *__s2)`
- `wchar_t * std::wcsrchr (wchar_t *__p, wchar_t __c)`
- `wchar_t * std::wcsstr (wchar_t *__s1, const wchar_t *__s2)`
- `wchar_t * std::wmemchr (wchar_t *__p, wchar_t __c, size_t __n)`

### 6.101.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `wchar.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cwchar](#).

## 6.102 `cwchar` File Reference

### Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

### 6.102.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cwchar](#).

## 6.103 `cwctype` File Reference

### Namespaces

- namespace [std](#)

### Defines

- `#define _GLIBCXX_CWCTYPE`

### 6.103.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `wctype.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cwctype](#).

## 6.104 `cwctype` File Reference

### Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

#### 6.104.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cwctype](#).

## 6.105 `cxxabi.h` File Reference

### Classes

- class [\\_\\_gnu\\_cxx::recursive\\_init\\_error](#)  
*Exception thrown by `__cxa_guard_acquire`.  
6.7[stmt.dcl]/4: If control re-enters the declaration (recursively) while the object is being initialized, the behavior is undefined.*

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [abi](#)

### Typedefs

- typedef `__cxa_cdtor_return_type(* __cxxabiv1::__cxa_cdtor_type)(void *)`

### Functions

- `int __cxxabiv1::__cxa_atexit (void(*) (void *), void *, void *)`
- `void __cxxabiv1::__cxa_bad_cast ()`
- `void __cxxabiv1::__cxa_bad_typeid ()`
- `std::type\_info * __cxxabiv1::__cxa_current_exception_type ()` `__attribute__((__pure__))`
- `char * __cxxabiv1::__cxa_demangle (const char * __mangled_name, char * __output_buffer, size_t * __length, int * __status)`
- `int __cxxabiv1::__cxa_finalize (void *)`

- void **\_\_cxxabiv1::\_\_cxa\_guard\_abort** (\_\_guard \*)
- int **\_\_cxxabiv1::\_\_cxa\_guard\_acquire** (\_\_guard \*)
- void **\_\_cxxabiv1::\_\_cxa\_guard\_release** (\_\_guard \*)
- void **\_\_cxxabiv1::\_\_cxa\_pure\_virtual** (void) \_\_attribute\_\_((\_\_noreturn\_\_))
- **\_\_cxa\_vec\_ctor\_return\_type** **\_\_cxxabiv1::\_\_cxa\_vec\_ctor** (void \*\_\_dest\_array, void \*\_\_src\_array, size\_t \_\_element\_count, size\_t \_\_element\_size, **\_\_cxa\_ctor\_return\_type**(\*\_\_constructor)(void \*, void \*), **\_\_cxa\_ctor\_type** \_\_destructor)
- void **\_\_cxxabiv1::\_\_cxa\_vec\_cleanup** (void \*\_\_array\_address, size\_t \_\_element\_count, size\_t \_\_s, **\_\_cxa\_ctor\_type** \_\_destructor)
- **\_\_cxa\_vec\_ctor\_return\_type** **\_\_cxxabiv1::\_\_cxa\_vec\_ctor** (void \*\_\_array\_address, size\_t \_\_element\_count, size\_t \_\_element\_size, **\_\_cxa\_ctor\_type** \_\_constructor, **\_\_cxa\_ctor\_type** \_\_destructor)
- void **\_\_cxxabiv1::\_\_cxa\_vec\_delete** (void \*\_\_array\_address, size\_t \_\_element\_size, size\_t \_\_padding\_size, **\_\_cxa\_ctor\_type** \_\_destructor)
- void **\_\_cxxabiv1::\_\_cxa\_vec\_delete2** (void \*\_\_array\_address, size\_t \_\_element\_size, size\_t \_\_padding\_size, **\_\_cxa\_ctor\_type** \_\_destructor, void(\*\_\_dealloc)(void \*))
- void **\_\_cxxabiv1::\_\_cxa\_vec\_delete3** (void \*\_\_array\_address, size\_t \_\_element\_size, size\_t \_\_padding\_size, **\_\_cxa\_ctor\_type** \_\_destructor, void(\*\_\_dealloc)(void \*, size\_t))
- void **\_\_cxxabiv1::\_\_cxa\_vec\_dtor** (void \*\_\_array\_address, size\_t \_\_element\_count, size\_t \_\_element\_size, **\_\_cxa\_ctor\_type** \_\_destructor)
- void \* **\_\_cxxabiv1::\_\_cxa\_vec\_new** (size\_t \_\_element\_count, size\_t \_\_element\_size, size\_t \_\_padding\_size, **\_\_cxa\_ctor\_type** \_\_constructor, **\_\_cxa\_ctor\_type** \_\_destructor)
- void \* **\_\_cxxabiv1::\_\_cxa\_vec\_new2** (size\_t \_\_element\_count, size\_t \_\_element\_size, size\_t \_\_padding\_size, **\_\_cxa\_ctor\_type** \_\_constructor, **\_\_cxa\_ctor\_type** \_\_destructor, void \*(\*\_\_alloc)(size\_t), void(\*\_\_dealloc)(void \*))
- void \* **\_\_cxxabiv1::\_\_cxa\_vec\_new3** (size\_t \_\_element\_count, size\_t \_\_element\_size, size\_t \_\_padding\_size, **\_\_cxa\_ctor\_type** \_\_constructor, **\_\_cxa\_ctor\_type** \_\_destructor, void \*(\*\_\_alloc)(size\_t), void(\*\_\_dealloc)(void \*, size\_t))
- void \* **\_\_cxxabiv1::\_\_dynamic\_cast** (const void \*\_\_src\_ptr, const **\_\_class\_type\_info** \*\_\_src\_type, const **\_\_class\_type\_info** \*\_\_dst\_type, ptrdiff\_t \_\_src2dst)

### 6.105.1 Detailed Description

The header provides an interface to the C++ ABI.

Definition in file [cxxabi.h](#).

## 6.106 `cxxabi_forced.h` File Reference

### Classes

- class `__cxxabiv1::__forced_unwind`  
*Thrown as part of forced unwinding.  
A magic placeholder class that can be caught by reference to recognize forced unwinding.*

### 6.106.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<cxxabi.h>`.

Definition in file `cxxabi_forced.h`.

## 6.107 `cxxabi_tweaks.h` File Reference

### Defines

- `#define _GLIBCXX_CXA_VEC_CTOR_RETURN(x)`
- `#define _GLIBCXX_GUARD_BIT`
- `#define _GLIBCXX_GUARD_PENDING_BIT`
- `#define _GLIBCXX_GUARD_SET(x)`
- `#define _GLIBCXX_GUARD_TEST(x)`
- `#define _GLIBCXX_GUARD_WAITING_BIT`

### Typedefs

- `typedef void __cxxabiv1::__cxa_ctor_return_type`
- `typedef void __cxxabiv1::__cxa_vec_ctor_return_type`

### Functions

- `__extension__ typedef int __guard __cxxabiv1::__attribute__ ((mode(__DI-  
_)))`

### 6.107.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<cxxabi.h>`.

Definition in file `cxxabi_tweaks.h`.

## 6.108 debug.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)
- namespace [std](#)
- namespace [std::\\_\\_debug](#)

### Defines

- #define [\\_\\_glibcxx\\_requires\\_cond](#)(\_Cond, \_Msg)
- #define [\\_\\_glibcxx\\_requires\\_heap](#)(\_First, \_Last)
- #define [\\_\\_glibcxx\\_requires\\_heap\\_pred](#)(\_First, \_Last, \_Pred)
- #define [\\_\\_glibcxx\\_requires\\_nonempty](#)()
- #define [\\_\\_glibcxx\\_requires\\_partitioned\\_lower](#)(\_First, \_Last, \_Value)
- #define [\\_\\_glibcxx\\_requires\\_partitioned\\_lower\\_pred](#)(\_First, \_Last, \_Value, \_Pred)
- #define [\\_\\_glibcxx\\_requires\\_partitioned\\_upper](#)(\_First, \_Last, \_Value)
- #define [\\_\\_glibcxx\\_requires\\_partitioned\\_upper\\_pred](#)(\_First, \_Last, \_Value, \_Pred)
- #define [\\_\\_glibcxx\\_requires\\_sorted](#)(\_First, \_Last)
- #define [\\_\\_glibcxx\\_requires\\_sorted\\_pred](#)(\_First, \_Last, \_Pred)
- #define [\\_\\_glibcxx\\_requires\\_sorted\\_set](#)(\_First1, \_Last1, \_First2)
- #define [\\_\\_glibcxx\\_requires\\_sorted\\_set\\_pred](#)(\_First1, \_Last1, \_First2, \_Pred)
- #define [\\_\\_glibcxx\\_requires\\_string](#)(\_String)
- #define [\\_\\_glibcxx\\_requires\\_string\\_len](#)(\_String, \_Len)
- #define [\\_\\_glibcxx\\_requires\\_subscript](#)(\_N)
- #define [\\_\\_glibcxx\\_requires\\_valid\\_range](#)(\_First, \_Last)
- #define [GLIBCXX\\_DEBUG\\_ASSERT](#)(\_Condition)
- #define [GLIBCXX\\_DEBUG\\_ONLY](#)(\_Statement)
- #define [GLIBCXX\\_DEBUG\\_PEDASSERT](#)(\_Condition)

### 6.108.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug.h](#).

## 6.109 debug\_allocator.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::debug\\_allocator](#)< \_Alloc >

*A meta-allocator with debugging bits, as per [20.4].  
This is precisely the allocator defined in the C++ Standard.*

- all allocation calls operator `new`
- all deallocation calls operator `delete`.

## Namespaces

- namespace `__gnu_cxx`

### 6.109.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file `debug_allocator.h`.

## 6.110 `debug_map_base.hpp` File Reference

### 6.110.1 Detailed Description

Contains a debug-mode base for all maps.

Definition in file `debug_map_base.hpp`.

## 6.111 decimal File Reference

### Classes

- class `std::decimal::decimal128`  
*3.2.4 Class `decimal128`.*
- class `std::decimal::decimal32`  
*3.2.2 Class `decimal32`.*
- class `std::decimal::decimal64`  
*3.2.3 Class `decimal64`.*

### Namespaces

- namespace `std`
- namespace `std::decimal`



## Defines

- `#define _DECLARE_DECIMAL128_COMPOUND_ASSIGNMENT(_Op)`
- `#define _DECLARE_DECIMAL32_COMPOUND_ASSIGNMENT(_Op)`
- `#define _DECLARE_DECIMAL64_COMPOUND_ASSIGNMENT(_Op)`
- `#define _DECLARE_DECIMAL_BINARY_OP_WITH_DEC(_Op, _T1, _T2, _T3)`
- `#define _DECLARE_DECIMAL_BINARY_OP_WITH_INT(_Op, _Tp)`
- `#define _DECLARE_DECIMAL_COMPARISON(_Op, _Tp)`
- `#define _GLIBCXX_USE_DECIMAL_`

## Functions

- `double std::decimal::decimal128_to_double (decimal128 __d)`
- `float std::decimal::decimal128_to_float (decimal128 __d)`
- `long double std::decimal::decimal128_to_long_double (decimal128 __d)`
- `long long std::decimal::decimal128_to_long_long (decimal128 __d)`
- `double std::decimal::decimal32_to_double (decimal32 __d)`
- `float std::decimal::decimal32_to_float (decimal32 __d)`
- `long double std::decimal::decimal32_to_long_double (decimal32 __d)`
- `long long std::decimal::decimal32_to_long_long (decimal32 __d)`
- `double std::decimal::decimal64_to_double (decimal64 __d)`
- `float std::decimal::decimal64_to_float (decimal64 __d)`
- `long double std::decimal::decimal64_to_long_double (decimal64 __d)`
- `long long std::decimal::decimal64_to_long_long (decimal64 __d)`
- `double std::decimal::decimal_to_double (decimal32 __d)`
- `double std::decimal::decimal_to_double (decimal64 __d)`
- `double std::decimal::decimal_to_double (decimal128 __d)`
- `float std::decimal::decimal_to_float (decimal32 __d)`
- `float std::decimal::decimal_to_float (decimal64 __d)`
- `float std::decimal::decimal_to_float (decimal128 __d)`
- `long double std::decimal::decimal_to_long_double (decimal32 __d)`
- `long double std::decimal::decimal_to_long_double (decimal64 __d)`
- `long double std::decimal::decimal_to_long_double (decimal128 __d)`
- `long long std::decimal::decimal_to_long_long (decimal32 __d)`
- `long long std::decimal::decimal_to_long_long (decimal64 __d)`
- `long long std::decimal::decimal_to_long_long (decimal128 __d)`
- `static decimal128 std::decimal::make_decimal128 (long long __coeff, int __exp)`
- `static decimal128 std::decimal::make_decimal128 (unsigned long long __coeff, int __exp)`
- `static decimal32 std::decimal::make_decimal32 (long long __coeff, int __exp)`

- static decimal32 **std::decimal::make\_decimal32** (unsigned long long \_\_coeff, int \_\_exp)
- static decimal64 **std::decimal::make\_decimal64** (unsigned long long \_\_coeff, int \_\_exp)
- static decimal64 **std::decimal::make\_decimal64** (long long \_\_coeff, int \_\_exp)
  
- bool **std::decimal::operator!=** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator!=** (decimal32 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator!=** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator!=** (decimal32 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator!=** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator!=** (decimal32 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator!=** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator!=** (int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator!=** (int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (long long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator!=** (decimal128 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator!=** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator!=** (decimal128 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator!=** (decimal128 \_\_lhs, unsigned long \_\_rhs)

- `bool std::decimal::operator!= (decimal128 __lhs, long long __rhs)`
- `bool std::decimal::operator!= (decimal128 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator!= (int __lhs, decimal128 __rhs)`
- `bool std::decimal::operator!= (unsigned int __lhs, decimal128 __rhs)`
- `bool std::decimal::operator!= (long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator!= (unsigned long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator!= (long long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator!= (unsigned long long __lhs, decimal128 __rhs)`
- `decimal32 std::decimal::operator* (decimal32 __lhs, unsigned int __rhs)`
- `decimal32 std::decimal::operator* (decimal32 __lhs, int __rhs)`
- `decimal32 std::decimal::operator* (decimal32 __lhs, unsigned long __rhs)`
- `decimal32 std::decimal::operator* (decimal32 __lhs, long __rhs)`
- `decimal32 std::decimal::operator* (decimal32 __lhs, long long __rhs)`
- `decimal32 std::decimal::operator* (decimal32 __lhs, unsigned long long __rhs)`
- `decimal32 std::decimal::operator* (int __lhs, decimal32 __rhs)`
- `decimal32 std::decimal::operator* (unsigned int __lhs, decimal32 __rhs)`
- `decimal32 std::decimal::operator* (long __lhs, decimal32 __rhs)`
- `decimal32 std::decimal::operator* (unsigned long __lhs, decimal32 __rhs)`
- `decimal32 std::decimal::operator* (long long __lhs, decimal32 __rhs)`
- `decimal32 std::decimal::operator* (unsigned long long __lhs, decimal32 __rhs)`
- `decimal64 std::decimal::operator* (decimal32 __lhs, decimal64 __rhs)`
- `decimal64 std::decimal::operator* (decimal64 __lhs, decimal32 __rhs)`
- `decimal64 std::decimal::operator* (decimal64 __lhs, decimal64 __rhs)`
- `decimal64 std::decimal::operator* (decimal64 __lhs, int __rhs)`
- `decimal64 std::decimal::operator* (decimal64 __lhs, unsigned int __rhs)`
- `decimal64 std::decimal::operator* (decimal64 __lhs, long __rhs)`
- `decimal64 std::decimal::operator* (decimal64 __lhs, unsigned long __rhs)`
- `decimal64 std::decimal::operator* (decimal64 __lhs, long long __rhs)`
- `decimal64 std::decimal::operator* (decimal64 __lhs, unsigned long long __rhs)`
- `decimal64 std::decimal::operator* (int __lhs, decimal64 __rhs)`
- `decimal64 std::decimal::operator* (unsigned int __lhs, decimal64 __rhs)`
- `decimal64 std::decimal::operator* (long __lhs, decimal64 __rhs)`
- `decimal64 std::decimal::operator* (unsigned long __lhs, decimal64 __rhs)`
- `decimal64 std::decimal::operator* (long long __lhs, decimal64 __rhs)`
- `decimal64 std::decimal::operator* (unsigned long long __lhs, decimal64 __rhs)`
- `decimal128 std::decimal::operator* (decimal32 __lhs, decimal128 __rhs)`
- `decimal128 std::decimal::operator* (decimal64 __lhs, decimal128 __rhs)`
- `decimal128 std::decimal::operator* (decimal128 __lhs, decimal32 __rhs)`
- `decimal128 std::decimal::operator* (decimal128 __lhs, decimal64 __rhs)`

- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, int \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, long \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, long long \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- decimal128 **std::decimal::operator\*** (int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (long long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- decimal32 **std::decimal::operator\*** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator+** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, int \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, long \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, long long \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- decimal128 **std::decimal::operator+** (int \_\_lhs, decimal128 \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, int \_\_rhs)
- decimal128 **std::decimal::operator+** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (long \_\_lhs, decimal128 \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- decimal128 **std::decimal::operator+** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (long long \_\_lhs, decimal128 \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, long \_\_rhs)
- decimal128 **std::decimal::operator+** (unsigned long long \_\_lhs, decimal128 \_\_rhs)

- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, long long \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- decimal32 **std::decimal::operator+** (int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator+** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator+** (long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator+** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator+** (long long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator+** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, int \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, long \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, long long \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- decimal64 **std::decimal::operator+** (int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator+** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator+** (long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator+** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_rhs)
- decimal64 **std::decimal::operator+** (long long \_\_lhs, decimal64 \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, int \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, long \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, long long \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- decimal32 **std::decimal::operator-** (int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator-** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator-** (long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator-** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator-** (long long \_\_lhs, decimal32 \_\_rhs)

- decimal32 **std::decimal::operator-** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, int \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, long \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, long long \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- decimal64 **std::decimal::operator-** (int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (long long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, int \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, long \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, long long \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- decimal128 **std::decimal::operator-** (int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (long long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, int \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, long \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, unsigned long \_\_rhs)

- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, long long \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- decimal32 **std::decimal::operator/** (int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator/** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator/** (long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator/** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- decimal128 **std::decimal::operator/** (long long \_\_lhs, decimal128 \_\_rhs)
- decimal32 **std::decimal::operator/** (long long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator/** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, int \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, long \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, long \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, long long \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- decimal64 **std::decimal::operator/** (int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (long long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, int \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, long long \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- decimal128 **std::decimal::operator/** (int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (long \_\_lhs, decimal128 \_\_rhs)

- decimal128 **std::decimal::operator/** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator<** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator<** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator<** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator<** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator<** (int \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (long long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (long \_\_lhs, decimal64 \_\_rhs)



- `bool std::decimal::operator< (decimal64 __lhs, long __rhs)`
- `bool std::decimal::operator< (decimal64 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator< (decimal128 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator< (decimal64 __lhs, long long __rhs)`
- `bool std::decimal::operator< (decimal64 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator== (int __lhs, decimal128 __rhs)`
- `bool std::decimal::operator== (unsigned int __lhs, decimal128 __rhs)`
- `bool std::decimal::operator== (long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator== (unsigned long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator== (long long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator== (unsigned long long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator== (decimal128 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator== (decimal32 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator== (decimal32 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator== (decimal128 __lhs, long long __rhs)`
- `bool std::decimal::operator== (decimal128 __lhs, long __rhs)`
- `bool std::decimal::operator== (decimal32 __lhs, long __rhs)`
- `bool std::decimal::operator== (decimal32 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator== (decimal64 __lhs, int __rhs)`
- `bool std::decimal::operator== (decimal128 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator== (long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator== (decimal128 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator== (decimal64 __lhs, long long __rhs)`
- `bool std::decimal::operator== (decimal64 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator== (decimal64 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator== (decimal64 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator== (long long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator== (unsigned long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator== (decimal128 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator== (long long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator== (decimal32 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator== (decimal32 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator== (unsigned long long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator== (decimal32 __lhs, int __rhs)`
- `bool std::decimal::operator== (decimal128 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator== (decimal32 __lhs, long long __rhs)`
- `bool std::decimal::operator== (decimal32 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator== (int __lhs, decimal32 __rhs)`
- `bool std::decimal::operator== (unsigned int __lhs, decimal32 __rhs)`
- `bool std::decimal::operator== (long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator== (decimal64 __lhs, long __rhs)`
- `bool std::decimal::operator== (decimal64 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator== (decimal64 __lhs, unsigned long __rhs)`

- `bool std::decimal::operator==(decimal64 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator==(int __lhs, decimal64 __rhs)`
- `bool std::decimal::operator==(unsigned int __lhs, decimal64 __rhs)`
- `bool std::decimal::operator==(unsigned long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator==(decimal128 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator==(decimal128 __lhs, int __rhs)`
- `bool std::decimal::operator==(unsigned long long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>(decimal32 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>(decimal64 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>(decimal64 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>(long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>(unsigned long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>(decimal128 __lhs, int __rhs)`
- `bool std::decimal::operator>(decimal32 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator>(decimal64 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator>(unsigned int __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>(int __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>(decimal32 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>(decimal32 __lhs, long long __rhs)`
- `bool std::decimal::operator>(decimal32 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator>(decimal32 __lhs, int __rhs)`
- `bool std::decimal::operator>(decimal32 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>(long long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>(unsigned long long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>(unsigned long long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>(decimal64 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>(long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>(int __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>(decimal32 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator>(unsigned long long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>(long long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>(decimal128 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator>(unsigned int __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>(decimal128 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>(decimal128 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator>(long long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>(decimal64 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator>(decimal128 __lhs, long __rhs)`
- `bool std::decimal::operator>(decimal64 __lhs, long __rhs)`
- `bool std::decimal::operator>(decimal128 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>(decimal128 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator>(decimal64 __lhs, int __rhs)`
- `bool std::decimal::operator>(decimal128 __lhs, long long __rhs)`

- `bool std::decimal::operator>` (decimal128 \_\_lhs, decimal64 \_\_rhs)
- `bool std::decimal::operator>` (unsigned long \_\_lhs, decimal128 \_\_rhs)
- `bool std::decimal::operator>` (decimal64 \_\_lhs, long long \_\_rhs)
- `bool std::decimal::operator>` (unsigned int \_\_lhs, decimal64 \_\_rhs)
- `bool std::decimal::operator>` (unsigned long \_\_lhs, decimal64 \_\_rhs)
- `bool std::decimal::operator>` (long \_\_lhs, decimal64 \_\_rhs)
- `bool std::decimal::operator>` (decimal32 \_\_lhs, long \_\_rhs)
- `bool std::decimal::operator>` (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- `bool std::decimal::operator>` (int \_\_lhs, decimal32 \_\_rhs)
- `bool std::decimal::operator>=` (unsigned long \_\_lhs, decimal32 \_\_rhs)
- `bool std::decimal::operator>=` (decimal64 \_\_lhs, int \_\_rhs)
- `bool std::decimal::operator>=` (decimal128 \_\_lhs, int \_\_rhs)
- `bool std::decimal::operator>=` (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- `bool std::decimal::operator>=` (decimal32 \_\_lhs, long \_\_rhs)
- `bool std::decimal::operator>=` (decimal32 \_\_lhs, decimal64 \_\_rhs)
- `bool std::decimal::operator>=` (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- `bool std::decimal::operator>=` (decimal64 \_\_lhs, unsigned long \_\_rhs)
- `bool std::decimal::operator>=` (decimal128 \_\_lhs, long \_\_rhs)
- `bool std::decimal::operator>=` (decimal32 \_\_lhs, long long \_\_rhs)
- `bool std::decimal::operator>=` (decimal64 \_\_lhs, unsigned int \_\_rhs)
- `bool std::decimal::operator>=` (long long \_\_lhs, decimal32 \_\_rhs)
- `bool std::decimal::operator>=` (decimal128 \_\_lhs, decimal128 \_\_rhs)
- `bool std::decimal::operator>=` (decimal64 \_\_lhs, decimal64 \_\_rhs)
- `bool std::decimal::operator>=` (decimal32 \_\_lhs, int \_\_rhs)
- `bool std::decimal::operator>=` (decimal64 \_\_lhs, long long \_\_rhs)
- `bool std::decimal::operator>=` (unsigned int \_\_lhs, decimal32 \_\_rhs)
- `bool std::decimal::operator>=` (long long \_\_lhs, decimal128 \_\_rhs)
- `bool std::decimal::operator>=` (decimal128 \_\_lhs, unsigned int \_\_rhs)
- `bool std::decimal::operator>=` (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- `bool std::decimal::operator>=` (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- `bool std::decimal::operator>=` (decimal128 \_\_lhs, unsigned long \_\_rhs)
- `bool std::decimal::operator>=` (decimal64 \_\_lhs, decimal32 \_\_rhs)
- `bool std::decimal::operator>=` (int \_\_lhs, decimal128 \_\_rhs)
- `bool std::decimal::operator>=` (decimal32 \_\_lhs, decimal32 \_\_rhs)
- `bool std::decimal::operator>=` (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- `bool std::decimal::operator>=` (decimal32 \_\_lhs, unsigned int \_\_rhs)
- `bool std::decimal::operator>=` (decimal128 \_\_lhs, decimal32 \_\_rhs)
- `bool std::decimal::operator>=` (unsigned long \_\_lhs, decimal64 \_\_rhs)
- `bool std::decimal::operator>=` (unsigned int \_\_lhs, decimal64 \_\_rhs)
- `bool std::decimal::operator>=` (decimal32 \_\_lhs, unsigned long \_\_rhs)
- `bool std::decimal::operator>=` (long \_\_lhs, decimal128 \_\_rhs)
- `bool std::decimal::operator>=` (int \_\_lhs, decimal64 \_\_rhs)
- `bool std::decimal::operator>=` (decimal32 \_\_lhs, decimal128 \_\_rhs)

- `bool std::decimal::operator>= (decimal128 __lhs, long long __rhs)`
- `bool std::decimal::operator>= (int __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>= (long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (unsigned long long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (long long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (unsigned int __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>= (long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, long __rhs)`
- `bool std::decimal::operator>= (unsigned long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, decimal64 __rhs)`

#### 6.111.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [decimal](#).

## 6.112 deque File Reference

#### 6.112.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [deque](#).

## 6.113 deque File Reference

#### Classes

- class [std::\\_\\_debug::deque< \\_Tp, \\_Allocator >](#)  
*Class [std::deque](#) with safety/checking/debug instrumentation.*

#### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_debug](#)

## Functions

- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator!= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator< (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator<= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator== (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator> (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator>= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`void std::__debug::swap (deque< _Tp, _Alloc > &__lhs, deque< _Tp, _Alloc > &__rhs)`

### 6.113.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/deque](#).

## 6.114 deque File Reference

### Classes

- class [std::\\_\\_profile::deque< \\_Tp, \\_Allocator >](#)  
*Class [std::deque](#) wrapper with performance instrumentation.*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_profile](#)

## Functions

- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator!= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator< (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator<= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator== (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator> (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator>= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`void std::__profile::swap (deque< _Tp, _Alloc > &__lhs, deque< _Tp, _Alloc > &__rhs)`

### 6.114.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/deque](#).

## 6.115 deque.tcc File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::copy (_Deque_iterator< _Tp, const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const _Tp &, const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`

- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::copy_backward (_Deque_iterator< _Tp, const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const _Tp &, const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`  
`void std::fill (const _Deque_iterator< _Tp, _Tp &, _Tp * > &__first, const _Deque_iterator< _Tp, _Tp &, _Tp * > &__last, const _Tp &__value)`
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::move (_Deque_iterator< _Tp, const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const _Tp &, const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::move_backward (_Deque_iterator< _Tp, const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const _Tp &, const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`

### 6.115.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<deque>`.

Definition in file [deque.tcc](#).

## 6.116 `enc_filebuf.h` File Reference

### Classes

- class [\\_\\_gnu\\_cxx::enc\\_filebuf< \\_CharT >](#)  
*class [enc\\_filebuf](#).*

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### 6.116.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [enc\\_filebuf.h](#).

## 6.117 `equally_split.h` File Reference

### Namespaces

- namespace `__gnu_parallel`

### Functions

- `template<typename _DifferenceType, typename _OutputIterator >  
_OutputIterator __gnu_parallel::equally_split (_DifferenceType __n, _-  
ThreadIndex __num_threads, _OutputIterator __s)`
- `template<typename _DifferenceType >  
_DifferenceType __gnu_parallel::equally_split_point (_DifferenceType __n, _-  
ThreadIndex __num_threads, _ThreadIndex __thread_no)`

#### 6.117.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file `equally_split.h`.

## 6.118 `error_constants.h` File Reference

### Namespaces

- namespace `std`

### Enumerations

- enum `errc` {  
    `address_family_not_supported`, `address_in_use`, `address_not_available`,  
    `already_connected`,  
    `argument_list_too_long`, `argument_out_of_domain`, `bad_address`, `bad_-`  
    `file_descriptor`,  
    `bad_message`, `broken_pipe`, `connection_aborted`, `connection_already_in_-`  
    `progress`,  
    `connection_refused`, `connection_reset`, `cross_device_link`, `destination_-`  
    `address_required`,  
    `device_or_resource_busy`, `directory_not_empty`, `executable_format_error`,  
    `file_exists`,  
    `file_too_large`, `filename_too_long`, `function_not_supported`, `host_-`  
    `unreachable`,



```

identifier_removed, illegal_byte_sequence, inappropriate_io_control_
operation, interrupted,
invalid_argument, invalid_seek, io_error, is_a_directory,
message_size, network_down, network_reset, network_unreachable,
no_buffer_space, no_child_process, no_link, no_lock_available,
no_message_available, no_message, no_protocol_option, no_space_on_
device,
no_stream_resources, no_such_device_or_address, no_such_device, no_
such_file_or_directory,
no_such_process, not_a_directory, not_a_socket, not_a_stream,
not_connected, not_enough_memory, not_supported, operation_canceled,
operation_in_progress, operation_not_permitted, operation_not_
supported, operation_would_block,
owner_dead, permission_denied, protocol_error, protocol_not_supported,
read_only_file_system, resource_deadlock_would_occur, resource_
unavailable_try_again, result_out_of_range,
state_not_recoverable, stream_timeout, text_file_busy, timed_out,
too_many_files_open_in_system, too_many_files_open, too_many_links,
too_many_symbolic_link_levels,
value_too_large, wrong_protocol_type }

```

### 6.118.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<system_error>`.

Definition in file [error\\_constants.h](#).

## 6.119 exception File Reference

### Classes

- class [std::bad\\_exception](#)
- class [std::exception](#)  
*Base class for all library exceptions.*

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

## Typedefs

- typedef void(\* [std::terminate\\_handler](#) )()
- typedef void(\* [std::unexpected\\_handler](#) )()

## Functions

- void [\\_\\_gnu\\_cxx::\\_\\_verbose\\_terminate\\_handler](#) ()
- terminate\_handler [std::set\\_terminate](#) (terminate\_handler) throw ()
- unexpected\_handler [std::set\\_unexpected](#) (unexpected\_handler) throw ()
- void [std::terminate](#) () \_\_attribute\_\_((\_\_noreturn\_\_)) throw ()
- bool [std::uncaught\\_exception](#) () \_\_attribute\_\_((\_\_pure\_\_)) throw ()
- void [std::unexpected](#) () \_\_attribute\_\_((\_\_noreturn\_\_))

### 6.119.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [exception](#).

## 6.120 exception.hpp File Reference

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Functions

- void [\\_\\_gnu\\_pbds::\\_\\_throw\\_container\\_error](#) (void)
- void [\\_\\_gnu\\_pbds::\\_\\_throw\\_insert\\_error](#) (void)
- void [\\_\\_gnu\\_pbds::\\_\\_throw\\_join\\_error](#) (void)
- void [\\_\\_gnu\\_pbds::\\_\\_throw\\_resize\\_error](#) (void)

### 6.120.1 Detailed Description

Contains exception classes.

Definition in file [exception.hpp](#).

## 6.121 exception\_defines.h File Reference

### Defines

- #define `__catch(X)`
- #define `__throw_exception_again`
- #define `__try`

### 6.121.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<exception>`.

Definition in file [exception\\_defines.h](#).

## 6.122 exception\_ptr.h File Reference

### Classes

- class [std::\\_\\_exception\\_ptr::exception\\_ptr](#)  
*An opaque pointer to an arbitrary exception.*

### Namespaces

- namespace [std](#)

### Functions

- template<typename \_Ex >  
exception\_ptr [std::copy\\_exception](#) (\_Ex \_\_ex) throw ()
- exception\_ptr [std::current\\_exception](#) () throw ()
- template<typename \_Ex >  
exception\_ptr [std::make\\_exception\\_ptr](#) (\_Ex \_\_ex) throw ()
- bool [std::\\_\\_exception\\_ptr::operator!=](#) (const exception\_ptr &, const exception\_ptr &) \_\_attribute\_\_((\_\_pure\_\_)) throw ()
- bool [std::\\_\\_exception\\_ptr::operator==](#) (const exception\_ptr &, const exception\_ptr &) \_\_attribute\_\_((\_\_pure\_\_)) throw ()
- void [std::rethrow\\_exception](#) (exception\_ptr) \_\_attribute\_\_((\_\_noreturn\_\_))
- void [std::\\_\\_exception\\_ptr::swap](#) (exception\_ptr &\_\_lhs, exception\_ptr &\_\_rhs)

### 6.122.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<exception_ptr.h>`.

Definition in file `exception_ptr.h`.

## 6.123 extc++.h File Reference

### 6.123.1 Detailed Description

This is an implementation file for a precompiled header.

Definition in file `extc++.h`.

## 6.124 extptr\_allocator.h File Reference

### Classes

- class `__gnu_cxx::_ExtPtr_allocator<_Tp>`

*An example allocator which uses a non-standard pointer type.*

*This allocator specifies that containers use a 'relative pointer' as it's pointer type. (See `ext/pointer.h`) Memory allocation in this example is still performed using `std::allocator`.*

### Namespaces

- namespace `__gnu_cxx`

### Functions

- `template<typename _Tp>  
void __gnu_cxx::swap (_ExtPtr_allocator<_Tp> &__larg, _ExtPtr_allocator<_Tp> &__rarg)`

### 6.124.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

### Author

Bob Walters

An example allocator which uses an alternative pointer type from `bits/pointer.h`. Supports test cases which confirm container support for alternative pointers.

Definition in file [extptr\\_allocator.h](#).

## 6.125 features.h File Reference

Defines on whether to include algorithm variants.

### Defines

- `#define _GLIBCXX_BAL_QUICKSORT`
- `#define _GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS`
- `#define _GLIBCXX_FIND_EQUAL_SPLIT`
- `#define _GLIBCXX_FIND_GROWING_BLOCKS`
- `#define _GLIBCXX_MERGESORT`
- `#define _GLIBCXX_QUICKSORT`
- `#define _GLIBCXX_TREE_DYNAMIC_BALANCING`
- `#define _GLIBCXX_TREE_FULL_COPY`
- `#define _GLIBCXX_TREE_INITIAL_SPLITTING`

### 6.125.1 Detailed Description

Defines on whether to include algorithm variants. Less variants reduce executable size and compile time. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [features.h](#).

### 6.125.2 Define Documentation

#### 6.125.2.1 `#define _GLIBCXX_BAL_QUICKSORT`

Include parallel dynamically load-balanced quicksort.

#### See also

`__gnu_parallel::_Settings::sort_algorithm`

Definition at line 55 of file `features.h`.

### 6.125.2.2 `#define _GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS`

Include the equal-sized blocks variant for `std::find`.

**See also**

`__gnu_parallel::_Settings::find_algorithm`

Definition at line 67 of file `features.h`.

### 6.125.2.3 `#define _GLIBCXX_FIND_EQUAL_SPLIT`

Include the equal splitting variant for `std::find`.

**See also**

`__gnu_parallel::_Settings::find_algorithm`

Definition at line 74 of file `features.h`.

### 6.125.2.4 `#define _GLIBCXX_FIND_GROWING_BLOCKS`

Include the growing blocks variant for `std::find`.

**See also**

`__gnu_parallel::_Settings::find_algorithm`

Definition at line 61 of file `features.h`.

### 6.125.2.5 `#define _GLIBCXX_MERGESORT`

Include parallel multi-way mergesort.

**See also**

`__gnu_parallel::_Settings::sort_algorithm`

Definition at line 41 of file `features.h`.

### 6.125.2.6 #define \_GLIBCXX\_QUICKSORT

Include parallel unbalanced quicksort.

**See also**

`__gnu_parallel::_Settings::sort_algorithm`

Definition at line 48 of file features.h.

### 6.125.2.7 #define \_GLIBCXX\_TREE\_DYNAMIC\_BALANCING

Include the dynamic balancing variant for `_Rb_tree::insert_unique(_Iter beg, _Iter __end)`.

**See also**

`__gnu_parallel::_Rb_tree`

Definition at line 91 of file features.h.

### 6.125.2.8 #define \_GLIBCXX\_TREE\_FULL\_COPY

In order to sort the input sequence of `_Rb_tree::insert_unique(_Iter beg, _Iter __end)` a full copy of the input elements is done.

**See also**

`__gnu_parallel::_Rb_tree`

Definition at line 100 of file features.h.

### 6.125.2.9 #define \_GLIBCXX\_TREE\_INITIAL\_SPLITTING

Include the initial splitting variant for `_Rb_tree::insert_unique(_Iter beg, _Iter __end)`.

**See also**

`__gnu_parallel::_Rb_tree`

Definition at line 83 of file features.h.

## 6.126 fenv.h File Reference

### 6.126.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [fenv.h](#).

## 6.127 find.h File Reference

Parallel implementation base for `std::find()`, `std::equal()` and related functions. This file is a GNU parallel extension to the Standard C++ Library.

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >`  
[std::pair](#)< [\\_RAIter1](#), [\\_RAIter2](#) > [\\_\\_gnu\\_parallel::\\_\\_find\\_template](#) ([\\_RAIter1](#)  
[\\_\\_begin1](#), [\\_RAIter1](#) [\\_\\_end1](#), [\\_RAIter2](#) [\\_\\_begin2](#), [\\_Pred](#) [\\_\\_pred](#), [\\_Selector](#) [\\_\\_-](#)  
[selector](#))
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >`  
[std::pair](#)< [\\_RAIter1](#), [\\_RAIter2](#) > [\\_\\_gnu\\_parallel::\\_\\_find\\_template](#) ([\\_RAIter1](#)  
[\\_\\_begin1](#), [\\_RAIter1](#) [\\_\\_end1](#), [\\_RAIter2](#) [\\_\\_begin2](#), [\\_Pred](#) [\\_\\_pred](#), [\\_Selector](#) [\\_\\_-](#)  
[selector](#), [constant\\_size\\_blocks\\_tag](#))
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >`  
[std::pair](#)< [\\_RAIter1](#), [\\_RAIter2](#) > [\\_\\_gnu\\_parallel::\\_\\_find\\_template](#) ([\\_RAIter1](#)  
[\\_\\_begin1](#), [\\_RAIter1](#) [\\_\\_end1](#), [\\_RAIter2](#) [\\_\\_begin2](#), [\\_Pred](#) [\\_\\_pred](#), [\\_Selector](#) [\\_\\_-](#)  
[selector](#), [growing\\_blocks\\_tag](#))
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >`  
[std::pair](#)< [\\_RAIter1](#), [\\_RAIter2](#) > [\\_\\_gnu\\_parallel::\\_\\_find\\_template](#) ([\\_RAIter1](#)  
[\\_\\_begin1](#), [\\_RAIter1](#) [\\_\\_end1](#), [\\_RAIter2](#) [\\_\\_begin2](#), [\\_Pred](#) [\\_\\_pred](#), [\\_Selector](#) [\\_\\_-](#)  
[selector](#), [equal\\_split\\_tag](#))

### 6.127.1 Detailed Description

Parallel implementation base for `std::find()`, `std::equal()` and related functions. This file is a GNU parallel extension to the Standard C++ Library.



Definition in file [find.h](#).

## 6.128 `find_selectors.h` File Reference

`_Function` objects representing different tasks to be plugged into the parallel find algorithm. This file is a GNU parallel extension to the Standard C++ Library.

### Classes

- struct [\\_\\_gnu\\_parallel::\\_\\_adjacent\\_find\\_selector](#)  
*Test predicate on two adjacent elements.*
- struct [\\_\\_gnu\\_parallel::\\_\\_find\\_first\\_of\\_selector<\\_FIterator>](#)  
*Test predicate on several elements.*
- struct [\\_\\_gnu\\_parallel::\\_\\_find\\_if\\_selector](#)  
*Test predicate on a single element, used for `std::find()` and `std::find_if()`.*
- struct [\\_\\_gnu\\_parallel::\\_\\_generic\\_find\\_selector](#)  
*Base class of all `__gnu_parallel::__find_template` selectors.*
- struct [\\_\\_gnu\\_parallel::\\_\\_mismatch\\_selector](#)  
*Test inverted predicate on a single element.*

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

#### 6.128.1 Detailed Description

`_Function` objects representing different tasks to be plugged into the parallel find algorithm. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [find\\_selectors.h](#).

## 6.129 `for_each.h` File Reference

Main interface for embarrassingly parallel functions.

## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

## Functions

- `template<typename _Iter, typename _UserOp, typename _Functionality, typename _Red, typename _Result >  
_UserOp __gnu_parallel::__for_each_template_random_access (_Iter __begin,  
_Iter __end, _UserOp __user_op, _Functionality &__functionality, _  
Red __reduction, _Result __reduction_start, _Result &__output, typename  
std::iterator_traits< _Iter >::difference_type __bound, _Parallelism __  
parallelism_tag)`

### 6.129.1 Detailed Description

Main interface for embarrassingly parallel functions. The explicit implementation are in other header files, like [workstealing.h](#), [par\\_loop.h](#), [omp\\_loop.h](#), and [omp\\_loop-static.h](#). This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [for\\_each.h](#).

## 6.130 for\_each\_selectors.h File Reference

Functors representing different tasks to be plugged into the generic parallelization methods for embarrassingly parallel functions. This file is a GNU parallel extension to the Standard C++ Library.

## Classes

- struct [\\_\\_gnu\\_parallel::\\_\\_accumulate\\_binop\\_reduct< \\_BinOp >](#)  
*General reduction, using a binary operator.*
- struct [\\_\\_gnu\\_parallel::\\_\\_accumulate\\_selector< \\_It >](#)  
*[std::accumulate\(\)](#) selector.*
- struct [\\_\\_gnu\\_parallel::\\_\\_adjacent\\_difference\\_selector< \\_It >](#)  
*Selector that returns the difference between two adjacent `__elements`.*
- struct [\\_\\_gnu\\_parallel::\\_\\_count\\_if\\_selector< \\_It, \\_Diff >](#)  
*[std::count\\_if\(\)](#) selector.*
- struct [\\_\\_gnu\\_parallel::\\_\\_count\\_selector< \\_It, \\_Diff >](#)

*std::count() selector.*

- struct `__gnu_parallel::__fill_selector< _It >`  
*std::fill() selector.*
- struct `__gnu_parallel::__for_each_selector< _It >`  
*std::for\_each() selector.*
- struct `__gnu_parallel::__generate_selector< _It >`  
*std::generate() selector.*
- struct `__gnu_parallel::__generic_for_each_selector< _It >`  
*Generic \_\_selector for embarrassingly parallel functions.*
- struct `__gnu_parallel::__identity_selector< _It >`  
*Selector that just returns the passed iterator.*
- struct `__gnu_parallel::__inner_product_selector< _It, _It2, _Tp >`  
*std::inner\_product() selector.*
- struct `__gnu_parallel::__max_element_reduct< _Compare, _It >`  
*Reduction for finding the maximum element, using a comparator.*
- struct `__gnu_parallel::__min_element_reduct< _Compare, _It >`  
*Reduction for finding the maximum element, using a comparator.*
- struct `__gnu_parallel::__replace_if_selector< _It, _Op, _Tp >`  
*std::replace() selector.*
- struct `__gnu_parallel::__replace_selector< _It, _Tp >`  
*std::replace() selector.*
- struct `__gnu_parallel::__transform1_selector< _It >`  
*std::transform() \_\_selector; one input sequence variant.*
- struct `__gnu_parallel::__transform2_selector< _It >`  
*std::transform() \_\_selector; two input sequences variant.*
- struct `__gnu_parallel::__DummyReduct`  
*Reduction function doing nothing.*
- struct `__gnu_parallel::__Nothing`  
*Functor doing nothing.*

## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### 6.130.1 Detailed Description

Functors representing different tasks to be plugged into the generic parallelization methods for embarrassingly parallel functions. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [for\\_each\\_selectors.h](#).

## 6.131 formatter.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)

### Enumerations

- enum `_Debug_msg_id` {  
`__msg_valid_range`, `__msg_insert_singular`, `__msg_insert_different`, `__msg_erase_bad`,  
`__msg_erase_different`, `__msg_subscript_oob`, `__msg_empty`, `__msg_unpartitioned`,  
`__msg_unpartitioned_pred`, `__msg_unsorted`, `__msg_unsorted_pred`, `__msg_not_heap`,  
`__msg_not_heap_pred`, `__msg_bad_bitset_write`, `__msg_bad_bitset_read`,  
`__msg_bad_bitset_flip`,  
`__msg_self_splice`, `__msg_splice_alloc`, `__msg_splice_bad`, `__msg_splice_other`,  
`__msg_splice_overlap`, `__msg_init_singular`, `__msg_init_copy_singular`, `__msg_init_const_singular`,  
`__msg_copy_singular`, `__msg_bad_deref`, `__msg_bad_inc`, `__msg_bad_dec`,  
`__msg_iter_subscript_oob`, `__msg_advance_oob`, `__msg_retreat_oob`, `__msg_iter_compare_bad`,  
`__msg_compare_different`, `__msg_iter_order_bad`, `__msg_order_different`,  
`__msg_distance_bad`,  
`__msg_distance_different`, `__msg_deref_istream`, `__msg_inc_istream`, `__msg_output_ostream`,

```
__msg_deref_istreambuf, __msg_inc_istreambuf, __msg_insert_after_end,
__msg_erase_after_bad,
__msg_valid_range2 }
```

## Functions

- `template<typename _Iterator >`  
`bool __gnu_debug::__check_singular (_Iterator &)`

### 6.131.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [formatter.h](#).

## 6.132 forward\_list File Reference

### 6.132.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [forward\\_list](#).

## 6.133 forward\_list File Reference

### Classes

- class `std::__debug::forward_list<_Tp, _Alloc >`  
*Class `std::forward_list` with safety/checking/debug instrumentation.*

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)
- namespace [std](#)
- namespace `std::__debug`

### Functions

- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator!= (const forward_list<_Tp, _Alloc > &__lx,`  
`const forward_list<_Tp, _Alloc > &__ly)`

- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator< (const forward_list< _Tp, _Alloc > &__lx,`  
`const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator<= (const forward_list< _Tp, _Alloc > &__lx,`  
`const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator== (const forward_list< _Tp, _Alloc > &__lx,`  
`const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator> (const forward_list< _Tp, _Alloc > &__lx, const`  
`forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator>= (const forward_list< _Tp, _Alloc > &__lx,`  
`const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`void std::__debug::swap (forward_list< _Tp, _Alloc > &__lx, forward_list<`  
`_Tp, _Alloc > &__ly)`

### 6.133.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/forward\\_list](#).

## 6.134 forward\_list File Reference

### Classes

- class [std::\\_\\_profile::forward\\_list< \\_Tp, \\_Alloc >](#)  
*Class [std::forward\\_list](#) wrapper with performance instrumentation.*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_profile](#)

### Functions

- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator!= (const forward_list< _Tp, _Alloc > &__lx,`  
`const forward_list< _Tp, _Alloc > &__ly)`

- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator< (const forward_list< _Tp, _Alloc > &__lx,`  
`const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator<= (const forward_list< _Tp, _Alloc > &__lx,`  
`const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator== (const forward_list< _Tp, _Alloc > &__lx,`  
`const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator> (const forward_list< _Tp, _Alloc > &__lx, const`  
`forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator>= (const forward_list< _Tp, _Alloc > &__lx,`  
`const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`void std::__profile::swap (forward_list< _Tp, _Alloc > &__lx, forward_list<`  
`_Tp, _Alloc > &__ly)`

### 6.134.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [profile/forward\\_list](#).

## 6.135 forward\_list.h File Reference

### Classes

- `struct std::_Fwd_list_base< _Tp, _Alloc >`  
*Base class for forward\_list.*
- `struct std::_Fwd_list_const_iterator< _Tp >`  
*A forward\_list::const\_iterator.*
- `struct std::_Fwd_list_iterator< _Tp >`  
*A forward\_list::iterator.*
- `struct std::_Fwd_list_node< _Tp >`  
*A helper node class for forward\_list. This is just a linked list with a data value in each node. There is a sorting utility method.*
- `struct std::_Fwd_list_node_base`

*A helper basic node class for forward\_list. This is just a linked list with nothing inside it. There are purely list shuffling utility methods here.*

- class [std::forward\\_list< \\_Tp, \\_Alloc >](#)

*A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.*

## Namespaces

- namespace [std](#)

## Functions

- `template<typename _Tp >`  
`bool std::operator!= (const _Fwd_list_iterator< _Tp > &__x, const _Fwd_list_iterator< _Tp > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator!= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator< (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator<= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp >`  
`bool std::operator== (const _Fwd_list_iterator< _Tp > &__x, const _Fwd_list_iterator< _Tp > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator> (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator>= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`void std::swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly)`



### 6.135.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<forward_list>`.

Definition in file [forward\\_list.h](#).

## 6.136 forward\_list.tcc File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _Tp, typename _Alloc >  
bool std::operator== (const forward_list< _Tp, _Alloc > &__lx, const  
forward_list< _Tp, _Alloc > &__ly)`

### 6.136.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<forward_list>`.

Definition in file [forward\\_list.tcc](#).

## 6.137 fstream File Reference

### Classes

- class [std::basic\\_filebuf< \\_CharT, \\_Traits >](#)  
*The actual work of input and output (for files).  
This class associates both its input and output sequence with an external disk file, and maintains a joint file position for both sequences. Many of its semantics are described in terms of similar behavior in the Standard C Library's FILE streams.*
- class [std::basic\\_fstream< \\_CharT, \\_Traits >](#)  
*Controlling input and output for files.  
This class supports reading from and writing to named files, using the inherited functions from [std::basic\\_istream](#). To control the associated sequence, an instance of [std::basic\\_filebuf](#) is used, which this page refers to as *sb*.*
- class [std::basic\\_ifstream< \\_CharT, \\_Traits >](#)

*Controlling input for files.*

*This class supports reading from named files, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.*

- class `std::basic_ofstream<_CharT, _Traits>`

*Controlling output for files.*

*This class supports reading from named files, using the inherited functions from `std::basic_ostream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.*

## Namespaces

- namespace `std`

### 6.137.1 Detailed Description

This is a Standard C++ Library header.

Definition in file `fstream`.

## 6.138 `fstream.tcc` File Reference

### Namespaces

- namespace `std`

### 6.138.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<fstream>`.

Definition in file `fstream.tcc`.

## 6.139 `functexcept.h` File Reference

### Namespaces

- namespace `std`

## Functions

- void **std::\_\_throw\_bad\_alloc** (void) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_bad\_cast** (void) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_bad\_exception** (void) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_bad\_function\_call** () \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_bad\_typeid** (void) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_domain\_error** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_future\_error** (int) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_invalid\_argument** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_ios\_failure** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_length\_error** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_logic\_error** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_out\_of\_range** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_overflow\_error** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_-  
))
- void **std::\_\_throw\_range\_error** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_runtime\_error** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_system\_error** (int) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_underflow\_error** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_-  
\_))

### 6.139.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include .

This header provides support for -fno-exceptions.

Definition in file [functexcept.h](#).

## 6.140 functional File Reference

### Classes

- struct [std::\\_\\_is\\_location\\_invariant< \\_Tp >](#)
- struct [std::\\_\\_Derives\\_from\\_binary\\_function< \\_Tp >](#)  
*Determines if the type `_Tp` derives from `binary_function`.*
- struct [std::\\_\\_Derives\\_from\\_unary\\_function< \\_Tp >](#)  
*Determines if the type `_Tp` derives from `unary_function`.*

- class `std::_Function_base`  
*Base class of all polymorphic function object wrappers.*
- struct `std::_Function_to_function_pointer< _Tp, _IsFunctionType >`  
*Turns a function type into a function pointer type.*
- struct `std::_Maybe_get_result_type< _Has_result_type, _Functor >`  
*If we have found a result\_type, extract it.*
- struct `std::_Maybe_unary_or_binary_function< _Res, _ArgTypes >`
- struct `std::_Maybe_unary_or_binary_function< _Res, _T1 >`  
*Derives from unary\_function, as appropriate.*
- struct `std::_Maybe_unary_or_binary_function< _Res, _T1, _T2 >`  
*Derives from binary\_function, as appropriate.*
- struct `std::_Maybe_wrap_member_pointer< _Tp >`
- struct `std::_Maybe_wrap_member_pointer< _Tp _Class::* >`
- class `std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) const >`  
*Implementation of mem\_fn for const member function pointers.*
- class `std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) const volatile >`  
*Implementation of mem\_fn for const volatile member function pointers.*
- class `std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) volatile >`  
*Implementation of mem\_fn for volatile member function pointers.*
- class `std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) >`  
*Implementation of mem\_fn for member function pointers.*
- class `std::_Mu< _Arg, false, false >`
- class `std::_Mu< _Arg, false, true >`
- class `std::_Mu< _Arg, true, false >`
- class `std::_Mu< reference_wrapper< _Tp >, false, false >`
- struct `std::_Placeholder< _Num >`  
*The type of placeholder objects defined by libstdc++.*
- struct `std::_Reference_wrapper_base< _Tp >`
- struct `std::_Safe_tuple_element< __i, _Tuple >`
- struct `std::_Safe_tuple_element_impl< __i, _Tuple, _IsSafe >`
- struct `std::_Safe_tuple_element_impl< __i, _Tuple, false >`

- struct `std::_Weak_result_type< _Functor >`
- struct `std::_Weak_result_type_impl< _Functor >`
- struct `std::_Weak_result_type_impl< _Res(&)(_ArgTypes...)>`  
*Retrieve the result type for a function reference.*
- struct `std::_Weak_result_type_impl< _Res(*)(_ArgTypes...)>`  
*Retrieve the result type for a function pointer.*
- struct `std::_Weak_result_type_impl< _Res(_ArgTypes...)>`  
*Retrieve the result type for a function type.*
- struct `std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) const >`  
*Retrieve result type for a const member function pointer.*
- struct `std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) const volatile >`  
*Retrieve result type for a const volatile member function pointer.*
- struct `std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) volatile >`  
*Retrieve result type for a volatile member function pointer.*
- struct `std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) >`  
*Retrieve result type for a member function pointer.*
- class `std::bad_function_call`  
*Exception class thrown when class template function's operator() is called with an empty target.*
- class `std::function< _Res(_ArgTypes...)>`  
*Primary class template for std::function.  
 Polymorphic function wrapper.*
- struct `std::is_bind_expression< _Tp >`  
*Determines if the given type \_Tp is a function object should be treated as a subexpression when evaluating calls to function objects returned by bind(). [TR1 3.6.1].*
- struct `std::is_bind_expression< _Bind< _Signature > >`  
*Class template \_Bind is always a bind expression.*
- struct `std::is_bind_expression< _Bind_result< _Result, _Signature > >`  
*Class template \_Bind is always a bind expression.*

- struct `std::is_placeholder<_Tp>`  
*Determines if the given type `_Tp` is a placeholder in a `bind()` expression and, if so, which placeholder it is. [TR1 3.6.2].*
- struct `std::is_placeholder<_Placeholder<_Num>>`
- class `std::reference_wrapper<_Tp>`  
*Primary class template for `reference_wrapper`.*

## Namespaces

- namespace `std`
- namespace `std::placeholders`

## Enumerations

- enum `_Manager_operation` { `__get_type_info`, `__get_functor_ptr`, `__clone_functor`, `__destroy_functor` }

## Functions

- template<typename \_Functor>  
`_Functor & std::__callable_functor (_Functor &__f)`
- template<typename \_Member, typename \_Class>  
`_Mem_fn<_Member _Class::*> std::__callable_functor (_Member _Class::*const &__p)`
- template<typename \_Member, typename \_Class>  
`_Mem_fn<_Member _Class::*> std::__callable_functor (_Member _Class::*&__p)`
- typename `_Tp` auto `std::__volget` (const volatile tuple< `_Tp...`> &\_\_tuple)-> typename tuple\_element< `_Ind`
- template<size\_t `_Ind`, typename... `_Tp`>  
auto `std::__volget` (volatile tuple< `_Tp...`> &\_\_tuple)-> typename tuple\_element< `_Ind`
- template<typename \_Functor, typename... \_ArgTypes>  
`_Bind_helper<_Functor, _ArgTypes...>::type std::bind (_Functor &&__f, _ArgTypes &&...__args)`
- template<typename \_Result, typename \_Functor, typename... \_ArgTypes>  
`_Bindres_helper<_Result, _Functor, _ArgTypes...>::type std::bind (_Functor &&__f, _ArgTypes &&...__args)`
- template<typename `_Tp`, typename `_Class`>  
`_Mem_fn<_Tp _Class::*> std::mem_fn (_Tp _Class::*__pm)`

- `template<typename _Res, typename... _Args>`  
`bool std::operator!= (const function< _Res(_Args...)> &__f, nullptr_t)`
- `template<typename _Res, typename... _Args>`  
`bool std::operator!= (nullptr_t, const function< _Res(_Args...)> &__f)`
- `template<typename _Res, typename... _Args>`  
`bool std::operator== (nullptr_t, const function< _Res(_Args...)> &__f)`
- `template<typename _Res, typename... _Args>`  
`bool std::operator== (const function< _Res(_Args...)> &__f, nullptr_t)`
- `template<typename _Res, typename... _Args>`  
`void std::swap (function< _Res(_Args...)> &__x, function< _Res(_Args...)> &__y)`
- `template<typename _Tp >`  
`reference_wrapper< _Tp > std::ref (_Tp &__t)`
- `template<typename _Tp >`  
`reference_wrapper< const _Tp > std::cref (const _Tp &__t)`
- `template<typename _Tp >`  
`reference_wrapper< _Tp > std::ref (reference_wrapper< _Tp > __t)`
- `template<typename _Tp >`  
`reference_wrapper< const _Tp > std::cref (reference_wrapper< _Tp > __t)`

## Variables

- `const _Placeholder< 1 > std::placeholders::_1`
- `const _Placeholder< 10 > std::placeholders::_10`
- `const _Placeholder< 11 > std::placeholders::_11`
- `const _Placeholder< 12 > std::placeholders::_12`
- `const _Placeholder< 13 > std::placeholders::_13`
- `const _Placeholder< 14 > std::placeholders::_14`
- `const _Placeholder< 15 > std::placeholders::_15`
- `const _Placeholder< 16 > std::placeholders::_16`
- `const _Placeholder< 17 > std::placeholders::_17`
- `const _Placeholder< 18 > std::placeholders::_18`
- `const _Placeholder< 19 > std::placeholders::_19`
- `const _Placeholder< 2 > std::placeholders::_2`
- `const _Placeholder< 20 > std::placeholders::_20`
- `const _Placeholder< 21 > std::placeholders::_21`
- `const _Placeholder< 22 > std::placeholders::_22`
- `const _Placeholder< 23 > std::placeholders::_23`
- `const _Placeholder< 24 > std::placeholders::_24`
- `const _Placeholder< 25 > std::placeholders::_25`
- `const _Placeholder< 26 > std::placeholders::_26`
- `const _Placeholder< 27 > std::placeholders::_27`

- `const _Placeholder< 28 > std::placeholders::_28`
- `const _Placeholder< 29 > std::placeholders::_29`
- `const _Placeholder< 3 > std::placeholders::_3`
- `const _Placeholder< 4 > std::placeholders::_4`
- `const _Placeholder< 5 > std::placeholders::_5`
- `const _Placeholder< 6 > std::placeholders::_6`
- `const _Placeholder< 7 > std::placeholders::_7`
- `const _Placeholder< 8 > std::placeholders::_8`
- `const _Placeholder< 9 > std::placeholders::_9`
- `enable_if< (!is_member_pointer< _Functor >::value &&!is_function< _Functor >::value &&!is_function< typename remove_pointer< _Functor >::type >::value), typename result_of< _Functor(_Args...)>::type >::type std::invoke(_Functor &_f, _Args &&...__args)`

#### 6.140.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [functional](#).

## 6.141 functional File Reference

### Classes

- class [\\_\\_gnu\\_cxx::binary\\_compose< \\_Operation1, \\_Operation2, \\_Operation3 >](#)  
*An SGI extension .*
- struct [\\_\\_gnu\\_cxx::constant\\_binary\\_fun< \\_Result, \\_Arg1, \\_Arg2 >](#)  
*An SGI extension .*
- struct [\\_\\_gnu\\_cxx::constant\\_unary\\_fun< \\_Result, \\_Argument >](#)  
*An SGI extension .*
- struct [\\_\\_gnu\\_cxx::constant\\_void\\_fun< \\_Result >](#)  
*An SGI extension .*
- struct [\\_\\_gnu\\_cxx::project1st< \\_Arg1, \\_Arg2 >](#)  
*An SGI extension .*
- struct [\\_\\_gnu\\_cxx::project2nd< \\_Arg1, \\_Arg2 >](#)  
*An SGI extension .*



- struct [\\_\\_gnu\\_cxx::select1st<\\_Pair>](#)  
*An SGI extension.*
- struct [\\_\\_gnu\\_cxx::select2nd<\\_Pair>](#)  
*An SGI extension.*
- class [\\_\\_gnu\\_cxx::subtractive\\_rng](#)
- class [\\_\\_gnu\\_cxx::unary\\_compose<\\_Operation1, \\_Operation2>](#)  
*An SGI extension.*

## Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

## Functions

- template<class \_Operation1, class \_Operation2 >  
unary\_compose<\_Operation1, \_Operation2 > [\\_\\_gnu\\_cxx::compose1](#) (const \_Operation1 &\_\_fn1, const \_Operation2 &\_\_fn2)
- template<class \_Operation1, class \_Operation2, class \_Operation3 >  
binary\_compose<\_Operation1, \_Operation2, \_Operation3 > [\\_\\_gnu\\_cxx::compose2](#) (const \_Operation1 &\_\_fn1, const \_Operation2 &\_\_fn2, const \_Operation3 &\_\_fn3)
- template<class \_Result >  
constant\_void\_fun<\_Result > [\\_\\_gnu\\_cxx::constant0](#) (const \_Result &\_\_val)
- template<class \_Result >  
constant\_unary\_fun<\_Result, \_Result > [\\_\\_gnu\\_cxx::constant1](#) (const \_Result &\_\_val)
- template<class \_Result >  
constant\_binary\_fun<\_Result, \_Result, \_Result > [\\_\\_gnu\\_cxx::constant2](#) (const \_Result &\_\_val)
- template<class \_Tp >  
\_Tp [\\_\\_gnu\\_cxx::identity\\_element](#) (std::multiplies<\_Tp>)
- template<class \_Tp >  
\_Tp [\\_\\_gnu\\_cxx::identity\\_element](#) (std::plus<\_Tp>)
- template<class \_Ret, class \_Tp, class \_Arg >  
mem\_fun1\_t<\_Ret, \_Tp, \_Arg > [\\_\\_gnu\\_cxx::mem\\_fun1](#) (\_Ret(\_Tp::\*\_\_f)(\_Arg))
- template<class \_Ret, class \_Tp, class \_Arg >  
mem\_fun1\_ref\_t<\_Ret, \_Tp, \_Arg > [\\_\\_gnu\\_cxx::mem\\_fun1\\_ref](#) (\_Ret(\_Tp::\*\_\_f)(\_Arg))

### 6.141.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [ext/functional](#).

## 6.142 functional\_hash.h File Reference

### Classes

- struct [std::hash< \\_Tp >](#)  
*Primary class template hash.*
- struct [std::hash< \\_Tp \\* >](#)  
*Partial specializations for pointer types.*

### Namespaces

- namespace [std](#)

### Defines

- `#define _Cxx_hashtable_define_trivial_hash(_Tp)`

### 6.142.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

Definition in file [functional\\_hash.h](#).

## 6.143 functions.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)

## Functions

- `template<typename _Iterator >`  
`bool \_\_gnu\_debug::\_\_check\_dereferenceable (_Iterator &)`
- `template<typename _Tp >`  
`bool \_\_gnu\_debug::\_\_check\_dereferenceable (const _Tp *__ptr)`
- `template<typename _Iterator, typename _Sequence >`  
`bool \_\_gnu\_debug::\_\_check\_dereferenceable (const _Safe_iterator< _Iterator, _Sequence > &__x)`
- `template<typename _ForwardIterator, typename _Tp >`  
`bool \_\_gnu\_debug::\_\_check\_partitioned\_lower (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _ForwardIterator, typename _Tp, typename _Pred >`  
`bool \_\_gnu\_debug::\_\_check\_partitioned\_lower (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value, _Pred __pred)`
- `template<typename _ForwardIterator, typename _Tp, typename _Pred >`  
`bool \_\_gnu\_debug::\_\_check\_partitioned\_upper (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value, _Pred __pred)`
- `template<typename _ForwardIterator, typename _Tp >`  
`bool \_\_gnu\_debug::\_\_check\_partitioned\_upper (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _Iterator >`  
`bool \_\_gnu\_debug::\_\_check\_singular (_Iterator &)`
- `template<typename _Iterator, typename _Sequence >`  
`bool \_\_gnu\_debug::\_\_check\_singular (const _Safe_iterator< _Iterator, _Sequence > &__x)`
- `template<typename _Tp >`  
`bool \_\_gnu\_debug::\_\_check\_singular (const _Tp *__ptr)`
- `bool \_\_gnu\_debug::\_\_check\_singular\_aux (const void *)`
- `template<typename _InputIterator >`  
`bool \_\_gnu\_debug::\_\_check\_sorted (const _InputIterator &__first, const _InputIterator &__last)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool \_\_gnu\_debug::\_\_check\_sorted (const _InputIterator &__first, const _InputIterator &__last, _Predicate __pred)`
- `template<typename _InputIterator >`  
`bool \_\_gnu\_debug::\_\_check\_sorted\_aux (const _InputIterator &, const _InputIterator &, std::input\_iterator\_tag)`
- `template<typename _ForwardIterator >`  
`bool \_\_gnu\_debug::\_\_check\_sorted\_aux (_ForwardIterator __first, _ForwardIterator __last, std::forward\_iterator\_tag)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool \_\_gnu\_debug::\_\_check\_sorted\_aux (const _InputIterator &, const _InputIterator &, _Predicate, std::input\_iterator\_tag)`

- `template<typename _ForwardIterator, typename _Predicate >`  
`bool __gnu_debug::__check_sorted_aux (_ForwardIterator __first, _-`  
`ForwardIterator __last, _Predicate __pred, std::forward\_iterator\_tag)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`bool __gnu_debug::__check_sorted_set (const _InputIterator1 &__first, const`  
`_InputIterator1 &__last, const _InputIterator2 &)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Predicate >`  
`bool __gnu_debug::__check_sorted_set (const _InputIterator1 &__first, const`  
`_InputIterator1 &__last, const _InputIterator2 &, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool __gnu_debug::__check_sorted_set_aux (const _InputIterator &__first,`  
`const _InputIterator &__last, _Predicate __pred, std::\_\_true\_type)`
- `template<typename _InputIterator >`  
`bool __gnu_debug::__check_sorted_set_aux (const _InputIterator &__first,`  
`const _InputIterator &__last, std::\_\_true\_type)`
- `template<typename _InputIterator >`  
`bool __gnu_debug::__check_sorted_set_aux (const _InputIterator &, const _-`  
`InputIterator &, std::\_\_false\_type)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool __gnu_debug::__check_sorted_set_aux (const _InputIterator &, const _-`  
`InputIterator &, _Predicate, std::\_\_false\_type)`
- `template<typename _CharT >`  
`const _CharT * \_\_gnu\_debug::\_\_check\_string (const _CharT *__s)`
- `template<typename _CharT, typename _Integer >`  
`const _CharT * \_\_gnu\_debug::\_\_check\_string (const _CharT *__s, const _-`  
`Integer &__n \_\_attribute\_\_\(\(\_\_unused\_\_\)\))`
- `template<typename _InputIterator >`  
`_InputIterator \_\_gnu\_debug::\_\_check\_valid\_range (const _InputIterator &__-`  
`first, const _InputIterator &__last \_\_attribute\_\_\(\(\_\_unused\_\_\)\))`
- `template<typename _Iterator, typename _Sequence >`  
`bool \_\_gnu\_debug::\_\_valid\_range (const _Safe_iterator< _Iterator, _Sequence`  
`> &__first, const _Safe_iterator< _Iterator, _Sequence > &__last)`
- `template<typename _InputIterator >`  
`bool \_\_gnu\_debug::\_\_valid\_range (const _InputIterator &__first, const _-`  
`InputIterator &__last)`
- `template<typename _InputIterator >`  
`bool \_\_gnu\_debug::\_\_valid\_range\_aux (const _InputIterator &__first, const _-`  
`InputIterator &__last, std::\_\_false\_type)`
- `template<typename _Integral >`  
`bool \_\_gnu\_debug::\_\_valid\_range\_aux (const _Integral &, const _Integral &,`  
`std::\_\_true\_type)`
- `template<typename _InputIterator >`  
`bool \_\_gnu\_debug::\_\_valid\_range\_aux2 (const _InputIterator &, const _-`  
`InputIterator &, std::input\_iterator\_tag)`

- `template<typename _RandomAccessIterator >`  
`bool __gnu_debug::__valid_range_aux2 (const _RandomAccessIterator &__-`  
`first, const _RandomAccessIterator &__last, std::random\_access\_iterator\_tag)`

### 6.143.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [functions.h](#).

## 6.144 future File Reference

### Classes

- class [std::\\_\\_basic\\_future< \\_Res >](#)  
*Common implementation for future and [shared\\_future](#).*
- struct [std::\\_\\_future\\_base](#)  
*Base class and enclosing scope.*
- struct [std::\\_\\_future\\_base::Ptr< \\_Res >](#)  
*A [unique\\_ptr](#) based on the instantiating type.*
- struct [std::\\_\\_future\\_base::Result< \\_Res >](#)  
*Result.*
- struct [std::\\_\\_future\\_base::Result< \\_Res & >](#)  
*Partial specialization for reference types.*
- struct [std::\\_\\_future\\_base::Result< void >](#)  
*Explicit specialization for void.*
- struct [std::\\_\\_future\\_base::Result\\_alloc< \\_Res, \\_Alloc >](#)  
*Result\_alloc.*
- struct [std::\\_\\_future\\_base::Result\\_base](#)  
*Base class for results.*
- class [std::\\_\\_future\\_base::State](#)  
*Shared state between a promise and one or more associated futures.*
- class [std::future< \\_Res >](#)

*Primary template for future.*

- class `std::future<_Res &>`  
*Partial specialization for `future<R&>`*
- class `std::future<void>`  
*Explicit specialization for `future<void>`*
- class `std::future_error`  
*Exception type thrown by futures.*
- struct `std::is_error_code_enum<future_errc>`  
*Specialization.*
- class `std::packaged_task<_Res(_ArgTypes...)>`  
*packaged\_task*
- class `std::promise<_Res>`  
*Primary template for promise.*
- class `std::promise<_Res &>`  
*Partial specialization for `promise<R&>`*
- class `std::promise<void>`  
*Explicit specialization for `promise<void>`*
- class `std::shared_future<_Res>`  
*Primary template for `shared_future`.*
- class `std::shared_future<_Res &>`  
*Partial specialization for `shared_future<R&>`*
- class `std::shared_future<void>`  
*Explicit specialization for `shared_future<void>`*

## Namespaces

- namespace `std`

## Enumerations

- enum `std::future_errc` { `broken_promise`, `future_already_retrieved`, `promise_already_satisfied`, `no_state` }
- enum `std::future_status` { `ready`, `timeout`, `deferred` }
- enum `std::launch` { `any`, `async`, `sync` }

## Functions

- template<typename `_Fn` , typename... `_Args`>  
future< typename result\_of< `_Fn`(`_Args`...)>::type > `std::async` (launch \_\_-  
policy, `_Fn` &&\_\_fn, `_Args` &&...\_\_args)
- template<typename `_Fn` , typename... `_Args`>  
enable\_if<!is\_same< typename decay< `_Fn` >::type, launch >::value, future<  
decltype(std::declval< `_Fn` >)(std::declval< `_Args` >)...)> >::type `std::async`  
(`_Fn` &&\_\_fn, `_Args` &&...\_\_args)
- const error\_category & `std::future_category` ()
- error\_code `std::make_error_code` (future\_errc \_\_errc)
- error\_condition `std::make_error_condition` (future\_errc \_\_errc)
- template<typename `_Res` >  
void `std::swap` (promise< `_Res` > &\_\_x, promise< `_Res` > &\_\_y)
- template<typename `_Res` , typename... `_ArgTypes`>  
void `std::swap` (packaged\_task< `_Res`(`_ArgTypes`...)> &\_\_x, packaged\_task<  
`_Res`(`_ArgTypes`...)> &\_\_y)

### 6.144.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [future](#).

## 6.145 `gslice.h` File Reference

### Classes

- class `std::gslice`  
*Class defining multi-dimensional subset of an array.*

### Namespaces

- namespace `std`

### 6.145.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

Definition in file [gslice.h](#).

## 6.146 `gslice_array.h` File Reference

### Classes

- class [std::gslice\\_array<\\_Tp>](#)  
*Reference to multi-dimensional subset of an array.*

### Namespaces

- namespace [std](#)

### Defines

- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`

### 6.146.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

Definition in file [gslice\\_array.h](#).

## 6.147 `hash_bytes.h` File Reference

### Namespaces

- namespace [std](#)

### Functions

- `size_t std::_Fnv_hash_bytes (const void *__ptr, size_t __len, size_t __seed)`
- `size_t std::_Hash_bytes (const void *__ptr, size_t __len, size_t __seed)`



### 6.147.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

Definition in file [hash\\_bytes.h](#).

## 6.148 `hash_fun.h` File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Functions

- `size_t __gnu_cxx::__stl_hash_string (const char *__s)`

### 6.148.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [hash\\_fun.h](#).

## 6.149 `hash_map` File Reference

### Classes

- class [\\_\\_gnu\\_cxx::hash\\_map<\\_Key, \\_Tp, \\_HashFn, \\_EqualKey, \\_Alloc>](#)
- class [\\_\\_gnu\\_cxx::hash\\_multimap<\\_Key, \\_Tp, \\_HashFn, \\_EqualKey, \\_Alloc>](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

### Functions

- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc>  
bool __gnu_cxx::operator!= (const hash_map<_Key, _Tp, _HashFn, _EqKey,  
_Alloc> &__hm1, const hash_map<_Key, _Tp, _HashFn, _EqKey, _Alloc>  
&__hm2)`

- `template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc >`  
`bool __gnu_cxx::operator!= (const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm1, const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm2)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >`  
`bool __gnu_cxx::operator== (const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc >`  
`bool __gnu_cxx::operator== (const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm1, const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm2)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >`  
`void __gnu_cxx::swap (hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >`  
`void __gnu_cxx::swap (hash_multimap< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, hash_multimap< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`

### 6.149.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [hash\\_map](#).

## 6.150 hash\_policy.hpp File Reference

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Defines

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`

- #define **PB\_DS\_CLASS\_C\_DEC**
- #define **PB\_DS\_CLASS\_T\_DEC**
- #define **PB\_DS\_CLASS\_T\_DEC**
- #define **PB\_DS\_CLASS\_T\_DEC**
- #define **PB\_DS\_CLASS\_T\_DEC**
- #define **PB\_DS\_CLASS\_T\_DEC**
- #define **PB\_DS\_CLASS\_T\_DEC**
- #define **PB\_DS\_CLASS\_T\_DEC**
- #define **PB\_DS\_CLASS\_T\_DEC**
- #define **PB\_DS\_CLASS\_T\_DEC**
- #define **PB\_DS\_SIZE\_BASE\_C\_DEC**

### 6.150.1 Detailed Description

Contains hash-related policies.

Definition in file [hash\\_policy.hpp](#).

## 6.151 hash\_set File Reference

### Classes

- class [\\_\\_gnu\\_cxx::hash\\_multiset<\\_Value, \\_HashFcn, \\_EqualKey, \\_Alloc>](#)
- class [\\_\\_gnu\\_cxx::hash\\_set<\\_Value, \\_HashFcn, \\_EqualKey, \\_Alloc>](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

### Functions

- template<class \_Value, class \_HashFcn, class \_EqualKey, class \_Alloc>  
bool **\_\_gnu\_cxx::operator!=** (const hash\_set< \_Value, \_HashFcn, \_EqualKey, \_Alloc> &\_\_hs1, const hash\_set< \_Value, \_HashFcn, \_EqualKey, \_Alloc> &\_\_hs2)
- template<class \_Val, class \_HashFcn, class \_EqualKey, class \_Alloc>  
bool **\_\_gnu\_cxx::operator!=** (const hash\_multiset< \_Val, \_HashFcn, \_EqualKey, \_Alloc> &\_\_hs1, const hash\_multiset< \_Val, \_HashFcn, \_EqualKey, \_Alloc> &\_\_hs2)

- `template<class _Value, class _HashFcn, class _EqualKey, class _Alloc >`  
`bool __gnu_cxx::operator== (const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`  
`bool __gnu_cxx::operator== (const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`  
`void __gnu_cxx::swap (hash_set< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, hash_set< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`  
`void __gnu_cxx::swap (hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`

### 6.151.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [hash\\_set](#).

## 6.152 hashtable.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Enumerations

- enum { `_S_num_primes` }

### Functions

- `unsigned long __gnu_cxx::__stl_next_prime (unsigned long __n)`
- `template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All >`  
`bool __gnu_cxx::operator!= (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht1, const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht2)`
- `template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All >`  
`bool __gnu_cxx::operator== (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht1, const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht2)`

- `template<class _Val, class _Key, class _HF, class _Extract, class _EqKey, class _All >  
void __gnu_cxx::swap (hashtable< _Val, _Key, _HF, _Extract, _EqKey, _All >  
&__ht1, hashtable< _Val, _Key, _HF, _Extract, _EqKey, _All > &__ht2)`

### Variables

- static const unsigned long **\_\_gnu\_cxx::\_\_stl\_prime\_list** [`_S_num_primes`]

#### 6.152.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [backward/hashtable.h](#).

## 6.153 **hashtable.h** File Reference

### Namespaces

- namespace [std](#)

#### 6.153.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<unordered_map>` or `<unordered_set>`.

Definition in file [bits/hashtable.h](#).

## 6.154 **hashtable\_policy.h** File Reference

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_detail](#)

### Functions

- `template<class _Iterator >  
std::iterator_traits< _Iterator >::difference_type std::__detail::__distance_fw  
( _Iterator __first, _Iterator __last, std::input\_iterator\_tag)`
- `template<class _Iterator >  
std::iterator_traits< _Iterator >::difference_type std::__detail::__distance_fw  
( _Iterator __first, _Iterator __last, std::forward\_iterator\_tag)`

- `template<class _Iterator >`  
`std::iterator_traits< _Iterator >::difference_type std::__detail::__distance_fw`  
`( _Iterator __first, _Iterator __last)`
- `template<typename _Value , bool __cache>`  
`bool std::__detail::operator!= (const _Node_iterator_base< _Value, __cache`  
`> &__x, const _Node_iterator_base< _Value, __cache > &__y)`
- `template<typename _Value , bool __cache>`  
`bool std::__detail::operator!= (const _Hashtable_iterator_base< _Value, __-`  
`cache > &__x, const _Hashtable_iterator_base< _Value, __cache > &__y)`
- `template<typename _Value , bool __cache>`  
`bool std::__detail::operator== (const _Hashtable_iterator_base< _Value, __-`  
`cache > &__x, const _Hashtable_iterator_base< _Value, __cache > &__y)`
- `template<typename _Value , bool __cache>`  
`bool std::__detail::operator== (const _Node_iterator_base< _Value, __cache`  
`> &__x, const _Node_iterator_base< _Value, __cache > &__y)`

## Variables

- `const unsigned long std::__detail::__prime_list [ ]`

### 6.154.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<unordered_map>` or `<unordered_set>`.

Definition in file [hashtable\\_policy.h](#).

## 6.155 indirect\_array.h File Reference

### Classes

- class [std::indirect\\_array< \\_Tp >](#)  
*Reference to arbitrary subset of an array.*

### Namespaces

- namespace [std](#)

### Defines

- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`

### 6.155.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

Definition in file [indirect\\_array.h](#).

## 6.156 `initializer_list` File Reference

### Classes

- class [std::initializer\\_list<\\_E>](#)  
*initializer\_list*

### Namespaces

- namespace [std](#)

### Functions

- `template<class _Tp>`  
`constexpr const _Tp * std::begin (initializer_list<_Tp> __ils)`
- `template<class _Tp>`  
`constexpr const _Tp * std::end (initializer_list<_Tp> __ils)`

### 6.156.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [initializer\\_list](#).

## 6.157 `ioomanip` File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _MoneyT>`  
`_Get_money<_MoneyT> std::get\_money (_MoneyT &__mon, bool __-intl=false)`

- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _`  
`CharT, _Traits > &__os, _Resetiosflags __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _`  
`CharT, _Traits > &__os, _Setbase __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _`  
`CharT, _Traits > &__os, _Setw __f)`
- `template<typename _CharT, typename _Traits, typename _MoneyT >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _`  
`CharT, _Traits > &__os, _Put_money< _MoneyT > __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _`  
`CharT, _Traits > &__os, _Setiosflags __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _`  
`CharT, _Traits > &__os, _Setfill< _CharT > __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _`  
`CharT, _Traits > &__os, _Setprecision __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _`  
`CharT, _Traits > &__is, _Setiosflags __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _`  
`CharT, _Traits > &__is, _Setw __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _`  
`CharT, _Traits > &__is, _Setbase __f)`
- `template<typename _CharT, typename _Traits, typename _MoneyT >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _`  
`CharT, _Traits > &__is, _Get_money< _MoneyT > __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _`  
`CharT, _Traits > &__is, _Setfill< _CharT > __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _`  
`CharT, _Traits > &__is, _Resetiosflags __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _`  
`CharT, _Traits > &__is, _Setprecision __f)`
- `template<typename _MoneyT >`  
`_Put_money< _MoneyT > std::put\_money (const _MoneyT &__mon, bool __`  
`intl=false)`



- `_Resetiosflags` [std::resetiosflags](#) (ios\_base::fmtflags \_\_mask)
- `_Setbase` [std::setbase](#) (int \_\_base)
- `template<typename _CharT >`  
  `_Setfill< _CharT >` [std::setfill](#) (\_CharT \_\_c)
- `_Setiosflags` [std::setiosflags](#) (ios\_base::fmtflags \_\_mask)
- `_Setprecision` [std::setprecision](#) (int \_\_n)
- `_Setw` [std::setw](#) (int \_\_n)

### 6.157.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [iomanip](#).

## 6.158 ios File Reference

### 6.158.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [ios](#).

## 6.159 ios\_base.h File Reference

### Classes

- class [std::ios\\_base](#)  
*The base of the I/O class hierarchy.*  
*This class defines everything that can be defined about I/O that does not depend on the type of characters being input or output. Most people will only see [ios\\_base](#) when they need to specify the full name of the various I/O flags (e.g., the openmodes).*
- class [std::ios\\_base::failure](#)  
*These are thrown to indicate problems with io.*  
*27.4.2.1.1 Class [ios\\_base::failure](#).*

### Namespaces

- namespace [std](#)

## Enumerations

- enum `_Ios_Fmtflags` {  
`_S_boolalpha`, `_S_dec`, `_S_fixed`, `_S_hex`,  
`_S_internal`, `_S_left`, `_S_oct`, `_S_right`,  
`_S_scientific`, `_S_showbase`, `_S_showpoint`, `_S_showpos`,  
`_S_skipws`, `_S_unitbuf`, `_S_uppercase`, `_S_adjustfield`,  
`_S_basefield`, `_S_floatfield`, `_S_ios_fmtflags_end` }
- enum `_Ios_Iostate` {  
`_S_goodbit`, `_S_badbit`, `_S_eofbit`, `_S_failbit`,  
`_S_ios_iostate_end` }
- enum `_Ios_Openmode` {  
`_S_app`, `_S_ate`, `_S_bin`, `_S_in`,  
`_S_out`, `_S_trunc`, `_S_ios_openmode_end` }
- enum `_Ios_Seekdir` { `_S_beg`, `_S_cur`, `_S_end`, `_S_ios_seekdir_end` }

## Functions

- `ios_base & std::boolalpha (ios_base &__base)`
- `ios_base & std::dec (ios_base &__base)`
- `ios_base & std::fixed (ios_base &__base)`
- `ios_base & std::hex (ios_base &__base)`
- `ios_base & std::internal (ios_base &__base)`
- `ios_base & std::left (ios_base &__base)`
- `ios_base & std::noboolalpha (ios_base &__base)`
- `ios_base & std::noshowbase (ios_base &__base)`
- `ios_base & std::noshowpoint (ios_base &__base)`
- `ios_base & std::noshowpos (ios_base &__base)`
- `ios_base & std::noskipws (ios_base &__base)`
- `ios_base & std::nounitbuf (ios_base &__base)`
- `ios_base & std::nouppercase (ios_base &__base)`
- `ios_base & std::oct (ios_base &__base)`
- `constexpr _Ios_Fmtflags std::operator& (_Ios_Fmtflags __a, _Ios_Fmtflags __b)`
- `constexpr _Ios_Openmode std::operator& (_Ios_Openmode __a, _Ios_Openmode __b)`
- `constexpr _Ios_Iostate std::operator& (_Ios_Iostate __a, _Ios_Iostate __b)`
- `const _Ios_Iostate & std::operator&= (_Ios_Iostate &__a, _Ios_Iostate __b)`
- `const _Ios_Openmode & std::operator&= (_Ios_Openmode &__a, _Ios_Openmode __b)`

- `const _Ios_Fmtflags & std::operator&= (_Ios_Fmtflags &__a, _Ios_Fmtflags __b)`
- `constexpr _Ios_Iostate std::operator^ (_Ios_Iostate __a, _Ios_Iostate __b)`
- `constexpr _Ios_Openmode std::operator^ (_Ios_Openmode __a, _Ios_Openmode __b)`
- `constexpr _Ios_Fmtflags std::operator^ (_Ios_Fmtflags __a, _Ios_Fmtflags __b)`
- `const _Ios_Iostate & std::operator^= (_Ios_Iostate &__a, _Ios_Iostate __b)`
- `const _Ios_Openmode & std::operator^= (_Ios_Openmode &__a, _Ios_Openmode __b)`
- `const _Ios_Fmtflags & std::operator^= (_Ios_Fmtflags &__a, _Ios_Fmtflags __b)`
- `constexpr _Ios_Fmtflags std::operator| (_Ios_Fmtflags __a, _Ios_Fmtflags __b)`
- `constexpr _Ios_Iostate std::operator| (_Ios_Iostate __a, _Ios_Iostate __b)`
- `constexpr _Ios_Openmode std::operator| (_Ios_Openmode __a, _Ios_Openmode __b)`
- `const _Ios_Fmtflags & std::operator|= (_Ios_Fmtflags &__a, _Ios_Fmtflags __b)`
- `const _Ios_Iostate & std::operator|= (_Ios_Iostate &__a, _Ios_Iostate __b)`
- `const _Ios_Openmode & std::operator|= (_Ios_Openmode &__a, _Ios_Openmode __b)`
- `constexpr _Ios_Fmtflags std::operator~ (_Ios_Fmtflags __a)`
- `constexpr _Ios_Iostate std::operator~ (_Ios_Iostate __a)`
- `constexpr _Ios_Openmode std::operator~ (_Ios_Openmode __a)`
- `ios_base & std::right (ios_base &__base)`
- `ios_base & std::scientific (ios_base &__base)`
- `ios_base & std::showbase (ios_base &__base)`
- `ios_base & std::showpoint (ios_base &__base)`
- `ios_base & std::showpos (ios_base &__base)`
- `ios_base & std::skipws (ios_base &__base)`
- `ios_base & std::unitbuf (ios_base &__base)`
- `ios_base & std::uppercase (ios_base &__base)`

### 6.159.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ios>`.

Definition in file [ios\\_base.h](#).

## 6.160 iosfwd File Reference

### Namespaces

- namespace [std](#)

### Typedefs

- typedef basic\_filebuf< char > [std::filebuf](#)
- typedef basic\_fstream< char > [std::fstream](#)
- typedef basic\_ifstream< char > [std::ifstream](#)
- typedef basic\_ios< char > [std::ios](#)
- typedef basic\_iostream< char > [std::iostream](#)
- typedef basic\_istream< char > [std::istream](#)
- typedef basic\_istreamstream< char > [std::istreamstream](#)
- typedef basic\_ofstream< char > [std::ofstream](#)
- typedef basic\_ostream< char > [std::ostream](#)
- typedef basic\_ostreamstream< char > [std::ostreamstream](#)
- typedef basic\_streambuf< char > [std::streambuf](#)
- typedef basic\_stringbuf< char > [std::stringbuf](#)
- typedef basic\_stringstream< char > [std::stringstream](#)
- typedef basic\_filebuf< wchar\_t > [std::wfilebuf](#)
- typedef basic\_fstream< wchar\_t > [std::wfstream](#)
- typedef basic\_ifstream< wchar\_t > [std::wifstream](#)
- typedef basic\_ios< wchar\_t > [std::wios](#)
- typedef basic\_iostream< wchar\_t > [std::wiostream](#)
- typedef basic\_istream< wchar\_t > [std::wistream](#)
- typedef basic\_istreamstream< wchar\_t > [std::wistreamstream](#)
- typedef basic\_ofstream< wchar\_t > [std::wofstream](#)
- typedef basic\_ostream< wchar\_t > [std::wostream](#)
- typedef basic\_ostreamstream< wchar\_t > [std::wostreamstream](#)
- typedef basic\_streambuf< wchar\_t > [std::wstreambuf](#)
- typedef basic\_stringbuf< wchar\_t > [std::wstringbuf](#)
- typedef basic\_stringstream< wchar\_t > [std::wstringstream](#)

### 6.160.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [iosfwd](#).

## 6.161 iostream File Reference

### Namespaces

- namespace [std](#)

### Variables

- static ios\_base::Init [std::\\_\\_ioinit](#)

### Standard Stream Objects

The `<iostream>` header declares the eight standard stream objects. For other declarations, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch24.html> and the *I/O forward declarations*

They are required by default to cooperate with the global C library's `FILE` streams, and to be available during program startup and termination. For more information, see the *HOWTO* linked to above.

- istream [std::cin](#)
- ostream [std::cout](#)
- ostream [std::cerr](#)
- ostream [std::clog](#)
- wistream [std::wcin](#)
- wostream [std::wcout](#)
- wostream [std::wcerr](#)
- wostream [std::wclog](#)

### 6.161.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [iostream](#).

## 6.162 istream File Reference

### Classes

- class [std::basic\\_istream<\\_CharT, \\_Traits>](#)  
*Merging istream and ostream capabilities.  
This class multiply inherits from the input and output stream classes simply to provide a single interface.*
- class [std::basic\\_istream<\\_CharT, \\_Traits>](#)

*Controlling input.*

*This is the base class for all input streams. It provides text formatting of all builtin types, and communicates with any class derived from [basic\\_streambuf](#) to do the actual input.*

- class [std::basic\\_istream<\\_CharT, \\_Traits>::sentry](#)

*Performs setup work for input streams.*

## Namespaces

- namespace [std](#)

## Functions

- `template<typename _CharT, typename _Traits, typename _Tp >  
basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &&__is, _Tp &__x)`
- `template<typename _CharT, typename _Traits >  
basic_istream< _CharT, _Traits > & std::ws (basic_istream< _CharT, _Traits > &__is)`
- `template<typename _CharT, typename _Traits >  
basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__in, _CharT &__c)`
- `template<class _Traits >  
basic_istream< char, _Traits > & std::operator>> (basic_istream< char, _Traits > &__in, unsigned char &__c)`
- `template<class _Traits >  
basic_istream< char, _Traits > & std::operator>> (basic_istream< char, _Traits > &__in, signed char &__c)`
- `template<typename _CharT, typename _Traits >  
basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__in, _CharT *__s)`
- `template<>  
basic_istream< char > & std::operator>> (basic_istream< char > &__in, char *__s)`
- `template<class _Traits >  
basic_istream< char, _Traits > & std::operator>> (basic_istream< char, _Traits > &__in, unsigned char *__s)`
- `template<class _Traits >  
basic_istream< char, _Traits > & std::operator>> (basic_istream< char, _Traits > &__in, signed char *__s)`

### 6.162.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [istream](#).

## 6.163 `istream.tcc` File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _CharT, typename _Traits >  
basic_istream< _CharT, _Traits > & std::ws (basic_istream< _CharT, _Traits  
> &__is)`
- `template<typename _CharT, typename _Traits >  
basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _-  
CharT, _Traits > &__in, _CharT &__c)`
- `template<typename _CharT, typename _Traits >  
basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _-  
CharT, _Traits > &__in, _CharT *__s)`

### 6.163.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<istream>`.

Definition in file [istream.tcc](#).

## 6.164 `iterator` File Reference

### 6.164.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [iterator](#).

## 6.165 iterator File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Functions

- `template<typename _InputIterator, typename _Distance >  
void \_\_gnu\_cxx::\_\_distance (_InputIterator __first, _InputIterator __last, _Distance &__n, std::input\_iterator\_tag)`
- `template<typename _RandomAccessIterator, typename _Distance >  
void \_\_gnu\_cxx::\_\_distance (_RandomAccessIterator __first, _RandomAccessIterator __last, _Distance &__n, std::random\_access\_iterator\_tag)`
- `template<typename _InputIterator, typename _Distance >  
void \_\_gnu\_cxx::distance (_InputIterator __first, _InputIterator __last, _Distance &__n)`

#### 6.165.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [ext/iterator](#).

## 6.166 iterator.h File Reference

Helper iterator classes for the `std::transform()` functions. This file is a GNU parallel extension to the Standard C++ Library.

### Classes

- class [\\_\\_gnu\\_parallel::\\_IteratorPair< \\_Iterator1, \\_Iterator2, \\_IteratorCategory >](#)  
*A pair of iterators. The usual iterator operations are applied to both child iterators.*
- class [\\_\\_gnu\\_parallel::\\_IteratorTriple< \\_Iterator1, \\_Iterator2, \\_Iterator3, \\_IteratorCategory >](#)  
*A triple of iterators. The usual iterator operations are applied to all three child iterators.*



## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### 6.166.1 Detailed Description

Helper iterator classes for the `std::transform()` functions. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [iterator.h](#).

## 6.167 iterator\_tracker.h File Reference

## Namespaces

- namespace [std](#)
- namespace [std::\\_\\_profile](#)

## Functions

- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool std::__profile::operator!= (const __iterator_tracker< _IteratorL, _-`  
`Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__`  
`rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool std::__profile::operator!= (const __iterator_tracker< _Iterator, _-`  
`Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`__iterator_tracker< _Iterator, _Sequence > std::__profile::operator+ (type-`  
`name __iterator_tracker< _Iterator, _Sequence >::difference_type __n, const`  
`__iterator_tracker< _Iterator, _Sequence > &__i)`
- `template<typename _Iterator, typename _Sequence >`  
`__iterator_tracker< _Iterator, _Sequence >::difference_type std::__-`  
`profile::operator- (const __iterator_tracker< _Iterator, _Sequence > &__lhs,`  
`const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`__iterator_tracker< _IteratorL, _Sequence >::difference_type std::__-`  
`profile::operator- (const __iterator_tracker< _IteratorL, _Sequence > &__lhs,`  
`const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool std::__profile::operator< (const __iterator_tracker< _IteratorL, _-`  
`Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__`  
`rhs)`

- `template<typename _Iterator, typename _Sequence >`  
`bool std::__profile::operator< (const __iterator_tracker< _Iterator, _Sequence`  
`> &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool std::__profile::operator<= (const __iterator_tracker< _Iterator, _`  
`Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool std::__profile::operator<= (const __iterator_tracker< _IteratorL, _`  
`Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__`  
`_rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool std::__profile::operator== (const __iterator_tracker< _Iterator, _`  
`Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool std::__profile::operator== (const __iterator_tracker< _IteratorL, _`  
`Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__`  
`_rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool std::__profile::operator> (const __iterator_tracker< _IteratorL, _`  
`Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__`  
`_rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool std::__profile::operator> (const __iterator_tracker< _Iterator, _Sequence`  
`> &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool std::__profile::operator>= (const __iterator_tracker< _Iterator, _`  
`Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool std::__profile::operator>= (const __iterator_tracker< _IteratorL, _`  
`Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__`  
`_rhs)`

### 6.167.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [iterator\\_tracker.h](#).

## 6.168 limits File Reference

### Classes

- struct `std::__numeric_limits_base`  
*Part of `std::numeric_limits`.*
- struct `std::numeric_limits< _Tp >`  
*Properties of fundamental types.*
- struct `std::numeric_limits< bool >`  
*`numeric_limits<bool>` specialization.*
- struct `std::numeric_limits< char >`  
*`numeric_limits<char>` specialization.*
- struct `std::numeric_limits< char16_t >`  
*`numeric_limits<char16_t>` specialization.*
- struct `std::numeric_limits< char32_t >`  
*`numeric_limits<char32_t>` specialization.*
- struct `std::numeric_limits< double >`  
*`numeric_limits<double>` specialization.*
- struct `std::numeric_limits< float >`  
*`numeric_limits<float>` specialization.*
- struct `std::numeric_limits< int >`  
*`numeric_limits<int>` specialization.*
- struct `std::numeric_limits< long >`  
*`numeric_limits<long>` specialization.*
- struct `std::numeric_limits< long double >`  
*`numeric_limits<long double>` specialization.*
- struct `std::numeric_limits< long long >`  
*`numeric_limits<long long>` specialization.*
- struct `std::numeric_limits< short >`  
*`numeric_limits<short>` specialization.*

- struct `std::numeric_limits< signed char >`  
*numeric\_limits<signed char> specialization.*
- struct `std::numeric_limits< unsigned char >`  
*numeric\_limits<unsigned char> specialization.*
- struct `std::numeric_limits< unsigned int >`  
*numeric\_limits<unsigned int> specialization.*
- struct `std::numeric_limits< unsigned long >`  
*numeric\_limits<unsigned long> specialization.*
- struct `std::numeric_limits< unsigned long long >`  
*numeric\_limits<unsigned long long> specialization.*
- struct `std::numeric_limits< unsigned short >`  
*numeric\_limits<unsigned short> specialization.*
- struct `std::numeric_limits< wchar_t >`  
*numeric\_limits<wchar\_t> specialization.*

## Namespaces

- namespace `std`

## Defines

- `#define __glibcxx_digits(T)`
- `#define __glibcxx_digits10(T)`
- `#define __glibcxx_double_has_denorm_loss`
- `#define __glibcxx_double_tinyness_before`
- `#define __glibcxx_double_traps`
- `#define __glibcxx_float_has_denorm_loss`
- `#define __glibcxx_float_tinyness_before`
- `#define __glibcxx_float_traps`
- `#define __glibcxx_integral_traps`
- `#define __glibcxx_long_double_has_denorm_loss`
- `#define __glibcxx_long_double_tinyness_before`
- `#define __glibcxx_long_double_traps`
- `#define __glibcxx_max(T)`
- `#define __glibcxx_max_digits10(T)`
- `#define __glibcxx_min(T)`
- `#define __glibcxx_signed(T)`

### Enumerations

- enum `std::float_denorm_style` { `std::denorm_indeterminate`, `std::denorm_absent`, `std::denorm_present` }
- enum `std::float_round_style` {  
    `round_indeterminate`, `std::round_toward_zero`, `std::round_to_nearest`,  
    `std::round_toward_infinity`,  
    `std::round_toward_neg_infinity` }

#### 6.168.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [limits](#).

### 6.169 list File Reference

#### 6.169.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [list](#).

### 6.170 list File Reference

#### Classes

- class `std::__debug::list<_Tp, _Allocator>`  
    Class `std::list` with safety/checking/debug instrumentation.

#### Namespaces

- namespace `std`
- namespace `std::__debug`

#### Functions

- template<typename \_Tp, typename \_Alloc>  
    bool **std::\_\_debug::operator!=** (const list<\_Tp, \_Alloc> &\_\_lhs, const list<\_Tp, \_Alloc> &\_\_rhs)

- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator< (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator<= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator== (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator> (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator>= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`void std::__debug::swap (list< _Tp, _Alloc > &__lhs, list< _Tp, _Alloc > &__rhs)`

### 6.170.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/list](#).

## 6.171 list File Reference

### Classes

- class [std::\\_\\_profile::list< \\_Tp, \\_Allocator >](#)  
*List wrapper with performance instrumentation.*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_profile](#)

### Functions

- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator!= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`

- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator< (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator<= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator== (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator> (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator>= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`void std::__profile::swap (list< _Tp, _Alloc > &__lhs, list< _Tp, _Alloc > &__rhs)`

### 6.171.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/list](#).

## 6.172 list.tcc File Reference

### Namespaces

- namespace [std](#)

### 6.172.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include

Definition in file [list.tcc](#).

## 6.173 list\_partition.h File Reference

\_\_Functionality to split \_\_sequence referenced by only input iterators. This file is a GNU parallel extension to the Standard C++ Library.

## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

## Functions

- `template<typename _Iter >`  
`void __gnu_parallel::__shrink (std::vector< _Iter > &__os_starts, size_t &__count_to_two, size_t &__range_length)`
- `template<typename _Iter >`  
`void __gnu_parallel::__shrink_and_double (std::vector< _Iter > &__os_starts, size_t &__count_to_two, size_t &__range_length, const bool __make_twice)`
- `template<typename _Iter, typename _FunctorType >`  
`size_t __gnu_parallel::list_partition (const _Iter __begin, const _Iter __end, _Iter *__starts, size_t *__lengths, const int __num_parts, _FunctorType &__f, int __oversampling=0)`

### 6.173.1 Detailed Description

\_\_Functionality to split \_\_sequence referenced by only input iterators. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [list\\_partition.h](#).

## 6.174 `list_update_policy.hpp` File Reference

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Defines

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`

### 6.174.1 Detailed Description

Contains policies for list update containers.

Definition in file [list\\_update\\_policy.hpp](#).



## 6.175 locale File Reference

### 6.175.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [locale](#).

## 6.176 locale\_classes.h File Reference

### Classes

- class [std::collate<\\_CharT>](#)  
*Facet for localized string comparison.*
- class [std::collate\\_byname<\\_CharT>](#)  
*class [collate\\_byname](#) [22.2.4.2].*
- class [std::locale](#)  
*Container class for localization functionality.  
The locale class is first a class wrapper for C library locales. It is also an extensible container for user-defined localization. A locale is a collection of facets that implement various localization features such as money, time, and number printing.*
- class [std::locale::facet](#)  
*Localization functionality base class.  
The facet class is the base class for a localization feature, such as money, time, and number printing. It provides common support for facets and reference management.*
- class [std::locale::id](#)  
*Facet ID class.  
The ID class provides facets with an index used to identify them. Every facet class must define a public static member [locale::id](#), or be derived from a facet that provides this member; otherwise the facet cannot be used in a locale. The [locale::id](#) ensures that each class type gets a unique identifier.*

### Namespaces

- namespace [std](#)

### Functions

- template<typename \_Facet>  
bool [std::has\\_facet](#) (const locale &\_\_loc) throw ()

- `template<typename _Facet >`  
`const _Facet & std::use\_facet (const locale &__loc)`

#### 6.176.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

Definition in file [locale\\_classes.h](#).

## 6.177 locale\_classes.tcc File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _Facet >`  
`bool std::has\_facet (const locale &__loc) throw ()`
- `template<typename _Facet >`  
`const _Facet & std::use\_facet (const locale &__loc)`

#### 6.177.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

Definition in file [locale\\_classes.tcc](#).

## 6.178 locale\_facets.h File Reference

### Classes

- class [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >](#)  
*Common base for ctype facet.*
- class [std::ctype< \\_CharT >](#)  
*Primary class template ctype facet.*  
*This template class defines classification and conversion functions for character sets.*  
*It wraps ctype functionality. Ctype gets used by streams for many I/O operations.*
- class [std::ctype< char >](#)

The `ctype<char>` specialization.

This class defines classification and conversion functions for the `char` type. It gets used by `char` streams for many I/O operations. The `char` specialization provides a number of optimizations as well.

- class `std::ctype<wchar_t>`

The `ctype<wchar_t>` specialization.

This class defines classification and conversion functions for the `wchar_t` type. It gets used by `wchar_t` streams for many I/O operations. The `wchar_t` specialization provides a number of optimizations as well.

- class `std::ctype_byname<_CharT>`

class `ctype_byname` [22.2.1.2].

- class `std::ctype_byname<char>`

22.2.1.4 Class `ctype_byname` specializations.

- class `std::num_get<_CharT, _InIter>`

Primary class template `num_get`.

This facet encapsulates the code to parse and return a number from a string. It is used by the `istream` numeric extraction operators.

- class `std::num_put<_CharT, _OutIter>`

Primary class template `num_put`.

This facet encapsulates the code to convert a number to a string. It is used by the `ostream` numeric insertion operators.

- class `std::numpunct<_CharT>`

Primary class template `numpunct`.

This facet stores several pieces of information related to printing and scanning numbers, such as the decimal point character. It takes a template parameter specifying the `char` type. The `numpunct` facet is used by streams for many I/O operations involving numbers.

- class `std::numpunct_byname<_CharT>`

class `numpunct_byname` [22.2.3.2].

## Namespaces

- namespace `std`

## Defines

- `#define _GLIBCXX_NUM_FACETS`

## Functions

- `template<typename _CharT >`  
`_CharT * std::__add_grouping (_CharT *__s, _CharT __sep, const char *__-`  
`gbeg, size_t __gsize, const _CharT *__first, const _CharT *__last)`
- `template<typename _Tp >`  
`void std::__convert_to_v (const char *, _Tp &, ios_base::iostate &, const __-`  
`c_locale &) throw ()`
- `template<>`  
`void std::__convert_to_v (const char *, long double &, ios_base::iostate &, const __-`  
`c_locale &) throw ()`
- `template<>`  
`void std::__convert_to_v (const char *, float &, ios_base::iostate &, const __-`  
`c_locale &) throw ()`
- `template<>`  
`void std::__convert_to_v (const char *, double &, ios_base::iostate &, const __-`  
`c_locale &) throw ()`
- `template<typename _CharT >`  
`ostreambuf_iterator< _CharT > std::__write (ostreambuf_iterator< _CharT >`  
`__s, const _CharT *__ws, int __len)`
- `template<typename _CharT, typename _OutIter >`  
`_OutIter std::__write (_OutIter __s, const _CharT *__ws, int __len)`
- `template<typename _CharT >`  
`bool std::isalnum (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isalpha (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::iscntrl (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isdigit (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isgraph (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::islower (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isprint (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::ispunct (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isspace (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isupper (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isxdigit (_CharT __c, const locale &__loc)`

- `template<typename _CharT >`  
`_CharT std::tolower (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`_CharT std::toupper (_CharT __c, const locale &__loc)`

### 6.178.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

Definition in file [locale\\_facets.h](#).

## 6.179 locale\_facets.tcc File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _CharT >`  
`_CharT * std::\_\_add\_grouping (_CharT *__s, _CharT __sep, const char *__-  
g beg, size_t __gsize, const _CharT *__first, const _CharT *__last)`
- `template<typename _CharT, typename _ValueT >`  
`int std::\_\_int\_to\_char (_CharT *__bufend, _ValueT __v, const _CharT *__lit,  
ios_base::fmtflags __flags, bool __dec)`
- `bool std::\_\_verify\_grouping (const char *__grouping, size_t __grouping_size,  
const string &__grouping_tmp) throw ()`

### 6.179.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

Definition in file [locale\\_facets.tcc](#).

## 6.180 locale\_facets\_nonio.h File Reference

### Classes

- class [std::messages<\\_CharT>](#)

Primary class template messages.

This facet encapsulates the code to retrieve messages from message catalogs. The only thing defined by the standard for this facet is the interface. All underlying functionality is implementation-defined.

- struct [std::messages\\_base](#)

Messages facet base class providing catalog typedef.

- class [std::messages\\_byname< \\_CharT >](#)

class [messages\\_byname](#) [22.2.7.2].

- class [std::money\\_base](#)

Money format ordering data.

This class contains an ordered array of 4 fields to represent the pattern for formatting a money amount. Each field may contain one entry from the part enum. symbol, sign, and value must be present and the remaining field must contain either none or space.

- class [std::money\\_get< \\_CharT, \\_InIter >](#)

Primary class template [money\\_get](#).

This facet encapsulates the code to parse and return a monetary amount from a string.

- class [std::money\\_put< \\_CharT, \\_OutIter >](#)

Primary class template [money\\_put](#).

This facet encapsulates the code to format and output a monetary amount.

- class [std::moneypunct< \\_CharT, \\_Intl >](#)

Primary class template [moneypunct](#).

This facet encapsulates the punctuation, grouping and other formatting features of money amount string representations.

- class [std::moneypunct\\_byname< \\_CharT, \\_Intl >](#)

class [moneypunct\\_byname](#) [22.2.6.4].

- class [std::time\\_base](#)

Time format ordering data.

This class provides an enum representing different orderings of time: day, month, and year.

- class [std::time\\_get< \\_CharT, \\_InIter >](#)

Primary class template [time\\_get](#).

This facet encapsulates the code to parse and return a date or time from a string. It is used by the istream numeric extraction operators.

- class [std::time\\_get\\_byname< \\_CharT, \\_InIter >](#)

class [time\\_get\\_byname](#) [22.2.5.2].

- class [std::time\\_put<\\_CharT, \\_OutIter >](#)  
*Primary class template [time\\_put](#).  
This facet encapsulates the code to format and output dates and times according to formats used by [strftime\(\)](#).*
- class [std::time\\_put\\_byname<\\_CharT, \\_OutIter >](#)  
*class [time\\_put\\_byname](#) [22.2.5.4].*

## Namespaces

- namespace [std](#)

### 6.180.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

Definition in file [locale\\_facets\\_nonio.h](#).

## 6.181 locale\_facets\_nonio.tcc File Reference

## Namespaces

- namespace [std](#)

### 6.181.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

Definition in file [locale\\_facets\\_nonio.tcc](#).

## 6.182 localefwd.h File Reference

## Namespaces

- namespace [std](#)

## Functions

- `template<typename _Facet >`  
`bool std::has_facet (const locale &__loc) throw ()`
- `template<typename _CharT >`  
`bool std::isalnum (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isalpha (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::iscntrl (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isdigit (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isgraph (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::islower (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isprint (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::ispunct (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isspace (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isupper (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isxdigit (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`_CharT std::tolower (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`_CharT std::toupper (_CharT __c, const locale &__loc)`
- `template<typename _Facet >`  
`const _Facet & std::use_facet (const locale &__loc)`

### 6.182.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

Definition in file [localefwd.h](#).

## 6.183 losertree.h File Reference

Many generic loser tree variants. This file is a GNU parallel extension to the Standard C++ Library.



## Classes

- class `__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >`  
*Stable `_LoserTree` variant.*
- class `__gnu_parallel::_LoserTree< false, _Tp, _Compare >`  
*Unstable `_LoserTree` variant.*
- class `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >`  
*Guarded loser/tournament tree.*
- struct `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser`  
*Internal representation of a `_LoserTree` element.*
- class `__gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >`  
*Stable `_LoserTree` implementation.*
- class `__gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >`  
*Unstable `_LoserTree` implementation.*
- class `__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >`  
*Base class of `_LoserTree` implementation using pointers.*
- struct `__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser`  
*Internal representation of `_LoserTree` \_\_elements.*
- class `__gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >`  
*Stable unguarded `_LoserTree` variant storing pointers.*
- class `__gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >`  
*Unstable unguarded `_LoserTree` variant storing pointers.*
- class `__gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare >`  
*Unguarded loser tree, keeping only pointers to the elements in the tree structure.*
- class `__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >`  
*Stable implementation of unguarded `_LoserTree`.*
- class `__gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >`  
*Non-Stable implementation of unguarded `_LoserTree`.*
- class `__gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare >`  
*Base class for unguarded `_LoserTree` implementation.*

## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### 6.183.1 Detailed Description

Many generic loser tree variants. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [losertree.h](#).

## 6.184 macros.h File Reference

### Defines

- #define [\\_\\_glibcxx\\_check\\_erase](#)(\_Position)
- #define [\\_\\_glibcxx\\_check\\_erase\\_after](#)(\_Position)
- #define [\\_\\_glibcxx\\_check\\_erase\\_range](#)(\_First, \_Last)
- #define [\\_\\_glibcxx\\_check\\_erase\\_range\\_after](#)(\_First, \_Last)
- #define [\\_\\_glibcxx\\_check\\_heap](#)(\_First, \_Last)
- #define [\\_\\_glibcxx\\_check\\_heap\\_pred](#)(\_First, \_Last, \_Pred)
- #define [\\_\\_glibcxx\\_check\\_insert](#)(\_Position)
- #define [\\_\\_glibcxx\\_check\\_insert\\_after](#)(\_Position)
- #define [\\_\\_glibcxx\\_check\\_insert\\_range](#)(\_Position, \_First, \_Last)
- #define [\\_\\_glibcxx\\_check\\_insert\\_range\\_after](#)(\_Position, \_First, \_Last)
- #define [\\_\\_glibcxx\\_check\\_nonempty](#)()
- #define [\\_\\_glibcxx\\_check\\_partitioned\\_lower](#)(\_First, \_Last, \_Value)
- #define [\\_\\_glibcxx\\_check\\_partitioned\\_lower\\_pred](#)(\_First, \_Last, \_Value, \_Pred)
- #define [\\_\\_glibcxx\\_check\\_partitioned\\_upper](#)(\_First, \_Last, \_Value)
- #define [\\_\\_glibcxx\\_check\\_partitioned\\_upper\\_pred](#)(\_First, \_Last, \_Value, \_Pred)
- #define [\\_\\_glibcxx\\_check\\_sorted](#)(\_First, \_Last)
- #define [\\_\\_glibcxx\\_check\\_sorted\\_pred](#)(\_First, \_Last, \_Pred)
- #define [\\_\\_glibcxx\\_check\\_sorted\\_set](#)(\_First1, \_Last1, \_First2)
- #define [\\_\\_glibcxx\\_check\\_sorted\\_set\\_pred](#)(\_First1, \_Last1, \_First2, \_Pred)
- #define [\\_\\_glibcxx\\_check\\_string](#)(\_String)
- #define [\\_\\_glibcxx\\_check\\_string\\_len](#)(\_String, \_Len)
- #define [\\_\\_glibcxx\\_check\\_subscript](#)(\_N)
- #define [\\_\\_glibcxx\\_check\\_valid\\_range](#)(\_First, \_Last)
- #define [\\_\\_GLIBCXX\\_DEBUG\\_VERIFY](#)(\_Condition, \_ErrorMessage)

### 6.184.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [macros.h](#).

### 6.184.2 Define Documentation

#### 6.184.2.1 `#define __glibcxx_check_erase( _Position )`

Verify that we can erase the element referenced by the iterator `_Position`. We can erase the element if the `_Position` iterator is dereferenceable and references this sequence.

Definition at line 126 of file `macros.h`.

#### 6.184.2.2 `#define __glibcxx_check_erase_after( _Position )`

Verify that we can erase the element after the iterator `_Position`. We can erase the element if the `_Position` iterator is before a dereferenceable one and references this sequence.

Definition at line 140 of file `macros.h`.

#### 6.184.2.3 `#define __glibcxx_check_erase_range( _First, _Last )`

Verify that we can erase the elements in the iterator range `[_First, _Last)`. We can erase the elements if `[_First, _Last)` is a valid iterator range within this sequence.

Definition at line 154 of file `macros.h`.

#### 6.184.2.4 `#define __glibcxx_check_erase_range_after( _First, _Last )`

Verify that we can erase the elements in the iterator range `(_First, _Last)`. We can erase the elements if `(_First, _Last)` is a valid iterator range within this sequence.

Definition at line 166 of file `macros.h`.

#### 6.184.2.5 `#define __glibcxx_check_heap_pred( _First, _Last, _Pred )`

Verify that the iterator range `[_First, _Last)` is a heap w.r.t. the predicate `_Pred`.  
Definition at line 297 of file `macros.h`.

**6.184.2.6 #define \_\_glibcxx\_check\_insert( *\_Position* )**

Verify that we can insert into \*this with the iterator *\_Position*. Insertion into a container at a specific position requires that the iterator be nonsingular, either dereferenceable or past-the-end, and that it reference the sequence we are inserting into. Note that this macro is only valid when the container is a *\_Safe\_sequence* and the iterator is a *\_Safe\_iterator*.

Definition at line 64 of file macros.h.

**6.184.2.7 #define \_\_glibcxx\_check\_insert\_after( *\_Position* )**

Verify that we can insert into \*this after the iterator *\_Position*. Insertion into a container after a specific position requires that the iterator be nonsingular, either dereferenceable or before-begin, and that it reference the sequence we are inserting into. Note that this macro is only valid when the container is a *\_Safe\_sequence* and the iterator is a *\_Safe\_iterator*.

Definition at line 81 of file macros.h.

**6.184.2.8 #define \_\_glibcxx\_check\_insert\_range( *\_Position*, *\_First*, *\_Last* )**

Verify that we can insert the values in the iterator range [*\_First*, *\_Last*) into \*this with the iterator *\_Position*. Insertion into a container at a specific position requires that the iterator be nonsingular (i.e., either dereferenceable or past-the-end), that it reference the sequence we are inserting into, and that the iterator range [*\_First*, *\_Last*) is a valid (possibly empty) range. Note that this macro is only valid when the container is a *\_Safe\_sequence* and the iterator is a *\_Safe\_iterator*.

**Todo**

We would like to be able to check for noninterference of *\_Position* and the range [*\_First*, *\_Last*), but that can't (in general) be done.

Definition at line 101 of file macros.h.

**6.184.2.9 #define \_\_glibcxx\_check\_insert\_range\_after( *\_Position*, *\_First*, *\_Last* )**

Verify that we can insert the values in the iterator range [*\_First*, *\_Last*) into \*this after the iterator *\_Position*. Insertion into a container after a specific position requires that the iterator be nonsingular (i.e., either dereferenceable or past-the-end), that it reference the sequence we are inserting into, and that the iterator range [*\_First*, *\_Last*) is a valid (possibly empty) range. Note that this macro is only valid when the container is a *\_Safe\_sequence* and the iterator is a *\_Safe\_iterator*.

### Todo

We would like to be able to check for noninterference of `_Position` and the range `[_First, _Last)`, but that can't (in general) be done.

Definition at line 118 of file macros.h.

**6.184.2.10** `#define __glibcxx_check_partitioned_lower( _First, _Last, _Value )`

Verify that the iterator range `[_First, _Last)` is partitioned w.r.t. the value `_Value`.

Definition at line 245 of file macros.h.

**6.184.2.11** `#define __glibcxx_check_partitioned_lower_pred( _First, _Last, _Value, _Pred )`

Verify that the iterator range `[_First, _Last)` is partitioned w.r.t. the value `_Value` and predicate `_Pred`.

Definition at line 265 of file macros.h.

**6.184.2.12** `#define __glibcxx_check_partitioned_upper_pred( _First, _Last, _Value, _Pred )`

Verify that the iterator range `[_First, _Last)` is partitioned w.r.t. the value `_Value` and predicate `_Pred`.

Definition at line 277 of file macros.h.

**6.184.2.13** `#define __glibcxx_check_sorted_pred( _First, _Last, _Pred )`

Verify that the iterator range `[_First, _Last)` is sorted by the predicate `_Pred`.

Definition at line 216 of file macros.h.

**6.184.2.14** `#define _GLIBCXX_DEBUG_VERIFY( _Condition, _ErrorMessage )`

Macros used by the implementation to verify certain properties. These macros may only be used directly by the debug wrappers. Note that these are macros (instead of the more obviously *correct* choice of making them functions) because we need line and file information at the call site, to minimize the distance between the user error and where the error is reported.

Definition at line 42 of file macros.h.

Referenced by `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_Safe_iterator()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator*()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator++()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator--()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator->()`, and `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator=()`.

## 6.185 malloc\_allocator.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::malloc\\_allocator< \\_Tp >](#)

*An allocator that uses malloc.*

*This is precisely the allocator defined in the C++ Standard.*

- all allocation calls `malloc`
- all deallocation calls `free`.

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Functions

- `template<typename _Tp >`  
`bool __gnu_cxx::operator!= (const malloc_allocator< _Tp > &, const malloc_allocator< _Tp > &)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator== (const malloc_allocator< _Tp > &, const malloc_allocator< _Tp > &)`

#### 6.185.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [malloc\\_allocator.h](#).

## 6.186 map File Reference

### 6.186.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [map](#).

## 6.187 map File Reference

### 6.187.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/map](#).

## 6.188 map File Reference

### 6.188.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/map](#).

## 6.189 map.h File Reference

### Classes

- class [std::\\_\\_debug::map<\\_Key, \\_Tp, \\_Compare, \\_Allocator>](#)  
*Class [std::map](#) with safety/checking/debug instrumentation.*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_debug](#)

### Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>  
bool std::__debug::operator!= (const map< _Key, _Tp, _Compare, _Allocator  
> &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`

- `template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >`  
`bool std::__debug::operator< (const map< _Key, _Tp, _Compare, _Allocator`  
`> &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >`  
`bool std::__debug::operator<= (const map< _Key, _Tp, _Compare, _`  
`Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >`  
`bool std::__debug::operator== (const map< _Key, _Tp, _Compare, _`  
`Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >`  
`bool std::__debug::operator> (const map< _Key, _Tp, _Compare, _Allocator`  
`> &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >`  
`bool std::__debug::operator>= (const map< _Key, _Tp, _Compare, _`  
`Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >`  
`void std::__debug::swap (map< _Key, _Tp, _Compare, _Allocator > &__lhs,`  
`map< _Key, _Tp, _Compare, _Allocator > &__rhs)`

### 6.189.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/map.h](#).

## 6.190 map.h File Reference

### Classes

- class [std::\\_\\_profile::map< \\_Key, \\_Tp, \\_Compare, \\_Allocator >](#)  
*Class [std::map](#) wrapper with performance instrumentation.*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_profile](#)



## Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator!= (const map< _Key, _Tp, _Compare, _Allocator`  
`> &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator< (const map< _Key, _Tp, _Compare, _Allocator`  
`> &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator<= (const map< _Key, _Tp, _Compare, _`  
`Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator== (const map< _Key, _Tp, _Compare, _`  
`Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator> (const map< _Key, _Tp, _Compare, _Allocator`  
`> &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator>= (const map< _Key, _Tp, _Compare, _`  
`Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`void std::__profile::swap (map< _Key, _Tp, _Compare, _Allocator > &__lhs,`  
`map< _Key, _Tp, _Compare, _Allocator > &__rhs)`

### 6.190.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/map.h](#).

## 6.191 mask\_array.h File Reference

### Classes

- class [std::mask\\_array< \\_Tp >](#)  
*Reference to selected subset of an array.*

### Namespaces

- namespace [std](#)

**Defines**

- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`

**6.191.1 Detailed Description**

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

Definition in file [mask\\_array.h](#).

**6.192 memory File Reference****6.192.1 Detailed Description**

This is a Standard C++ Library header.

Definition in file [memory](#).

**6.193 memory File Reference****Classes**

- [struct \\_\\_gnu\\_cxx::temporary\\_buffer<\\_ForwardIterator, \\_Tp>](#)

**Namespaces**

- [namespace \\_\\_gnu\\_cxx](#)

**Functions**

- `template<typename _InputIter, typename _Size, typename _ForwardIter >  
pair< _InputIter, _ForwardIter > __gnu_cxx::__uninitialized_copy_n (_  
InputIter __first, _Size __count, _ForwardIter __result, std::input\_iterator\_tag)`
- `template<typename _RandomAccessIter, typename _Size, typename _ForwardIter >  
pair< _RandomAccessIter, _ForwardIter > __gnu_cxx::__uninitialized_  
copy_n (_RandomAccessIter __first, _Size __count, _ForwardIter __result,  
std::random\_access\_iterator\_tag)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter >  
pair< _InputIter, _ForwardIter > __gnu_cxx::__uninitialized_copy_n (_  
InputIter __first, _Size __count, _ForwardIter __result)`

- `template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Tp >`  
`pair< _InputIter, _ForwardIter > __gnu_cxx::__uninitialized_copy_n_a ( _`  
`InputIter __first, _Size __count, _ForwardIter __result, std::allocator< _Tp >)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Allocator >`  
`pair< _InputIter, _ForwardIter > __gnu_cxx::__uninitialized_copy_n_a ( _`  
`InputIter __first, _Size __count, _ForwardIter __result, _Allocator __alloc)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter >`  
`pair< _InputIter, _ForwardIter > \_\_gnu\_cxx::uninitialized\_copy\_n (_InputIter`  
`__first, _Size __count, _ForwardIter __result)`

### 6.193.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [ext/memory](#).

## 6.194 merge.h File Reference

Parallel implementation of `std::merge()`. This file is a GNU parallel extension to the Standard C++ Library.

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _`  
`DifferenceTp, typename _Compare >`  
`_OutputIterator \_\_gnu\_parallel::\_\_merge\_advance (_RAIter1 &__begin1, _`  
`RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __`  
`target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename`  
`_DifferenceTp, typename _Compare >`  
`_OutputIterator \_\_gnu\_parallel::\_\_merge\_advance\_movc (_RAIter1 &_`  
`__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _`  
`OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename`  
`_DifferenceTp, typename _Compare >`  
`_OutputIterator \_\_gnu\_parallel::\_\_merge\_advance\_usual (_RAIter1 &_`  
`__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _`  
`OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`

- `template<typename _RAIter1, typename _RAIter3, typename _Compare >`  
`_RAIter3 \_\_gnu\_parallel::\_\_parallel\_merge\_advance (_RAIter1 &__begin1, _-`  
`RAIter1 __end1, _RAIter1 &__begin2, _RAIter1 __end2, _RAIter3 __target,`  
`typename std::iterator_traits< _RAIter1 >::difference_type __max_length, _-`  
`Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _Compare >`  
`_RAIter3 \_\_gnu\_parallel::\_\_parallel\_merge\_advance (_RAIter1 &__begin1, _-`  
`RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _RAIter3 __target,`  
`typename std::iterator_traits< _RAIter1 >::difference_type __max_length, _-`  
`Compare __comp)`

#### 6.194.1 Detailed Description

Parallel implementation of `std::merge()`. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [merge.h](#).

## 6.195 messages\_members.h File Reference

### Namespaces

- namespace [std](#)

#### 6.195.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

Definition in file [messages\\_members.h](#).

## 6.196 move.h File Reference

### Namespaces

- namespace [std](#)

### Defines

- `#define \_GLIBCXX\_FORWARD(_Tp, __val)`
- `#define \_GLIBCXX\_MOVE(__val)`

## Functions

- `template<typename _Tp >`  
`_Tp * std::__addressof (_Tp &__r)`
- `template<typename _Tp >`  
`_Tp * std::addressof (_Tp &__r)`
- `template<typename _Tp >`  
`_Tp && std::forward (typename std::remove_reference< _Tp >::type &__t)`
- `template<typename _Tp >`  
`_Tp && std::forward (typename std::remove_reference< _Tp >::type &&__t)`
- `template<typename _Tp >`  
`std::remove_reference< _Tp >::type && std::move (_Tp &&__t)`
- `template<typename _Tp, size_t _Nm>`  
`void std::swap (_Tp(&)[_Nm], _Tp(&)[_Nm])`
- `template<typename _Tp >`  
`void std::swap (_Tp &__a, _Tp &__b)`

### 6.196.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<utility>`.

Definition in file [move.h](#).

## 6.197 mt\_allocator.h File Reference

### Classes

- `struct __gnu_cxx::__common_pool_policy< _PoolTp, _Thread >`  
*Policy for shared \_\_pool objects.*
- `class __gnu_cxx::__mt_alloc< _Tp, _Poolp >`  
*This is a fixed size (power of 2) allocator which - when compiled with thread support - will maintain one freelist per size per thread plus a global one. Steps are taken to limit the per thread freelist sizes (by returning excess back to the global list). Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt12ch32.html>.*
- `class __gnu_cxx::__mt_alloc_base< _Tp >`  
*Base class for \_Tp dependent member functions.*
- `struct __gnu_cxx::__per_type_pool_policy< _Tp, _PoolTp, _Thread >`  
*Policy for individual \_\_pool objects.*

- class [\\_\\_gnu\\_cxx::\\_\\_pool< false >](#)  
*Specialization for single thread.*
- class [\\_\\_gnu\\_cxx::\\_\\_pool< true >](#)  
*Specialization for thread enabled, via `gthreads.h`.*
- struct [\\_\\_gnu\\_cxx::\\_\\_pool\\_base](#)  
*Base class for pool object.*

**Namespaces**

- namespace [\\_\\_gnu\\_cxx](#)

**Defines**

- `#define __thread_default`

**Typedefs**

- typedef `void(* __gnu_cxx::__destroy_handler)(void *)`

**Functions**

- `template<typename _Tp, typename _Poolp >`  
`bool \_\_gnu\_cxx::operator!= (const __mt_alloc< _Tp, _Poolp > &, const __-`  
`mt_alloc< _Tp, _Poolp > &)`
- `template<typename _Tp, typename _Poolp >`  
`bool \_\_gnu\_cxx::operator== (const __mt_alloc< _Tp, _Poolp > &, const __-`  
`mt_alloc< _Tp, _Poolp > &)`

**6.197.1 Detailed Description**

This file is a GNU extension to the Standard C++ Library.

Definition in file [mt\\_allocator.h](#).

**6.198 multimap.h File Reference****Classes**

- class [std::\\_\\_debug::multimap< \\_Key, \\_Tp, \\_Compare, \\_Allocator >](#)

Class [`std::multimap`](#) with safety/checking/debug instrumentation.

## Namespaces

- namespace [`std`](#)
- namespace [`std::\_\_debug`](#)

## Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator!= (const multimap< _Key, _Tp, _Compare, _-`  
`Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &_-`  
`rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator< (const multimap< _Key, _Tp, _Compare, _-`  
`Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &_-`  
`rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator<= (const multimap< _Key, _Tp, _Compare, _-`  
`Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &_-`  
`rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator== (const multimap< _Key, _Tp, _Compare, _-`  
`Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &_-`  
`rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator> (const multimap< _Key, _Tp, _Compare, _-`  
`Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &_-`  
`rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator>= (const multimap< _Key, _Tp, _Compare, _-`  
`Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &_-`  
`rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`void std::__debug::swap (multimap< _Key, _Tp, _Compare, _Allocator > &_-`  
`lhs, multimap< _Key, _Tp, _Compare, _Allocator > &rhs)`

### 6.198.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/multimap.h](#).

## 6.199 `multimap.h` File Reference

### Classes

- class `std::__profile::multimap<_Key, _Tp, _Compare, _Allocator>`  
*Class `std::multimap` wrapper with performance instrumentation.*

### Namespaces

- namespace `std`
- namespace `std::__profile`

### Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>`  
`bool std::__profile::operator!= (const multimap<_Key, _Tp, _Compare, _Allocator> &__lhs, const multimap<_Key, _Tp, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>`  
`bool std::__profile::operator< (const multimap<_Key, _Tp, _Compare, _Allocator> &__lhs, const multimap<_Key, _Tp, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>`  
`bool std::__profile::operator<= (const multimap<_Key, _Tp, _Compare, _Allocator> &__lhs, const multimap<_Key, _Tp, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>`  
`bool std::__profile::operator== (const multimap<_Key, _Tp, _Compare, _Allocator> &__lhs, const multimap<_Key, _Tp, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>`  
`bool std::__profile::operator> (const multimap<_Key, _Tp, _Compare, _Allocator> &__lhs, const multimap<_Key, _Tp, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>`  
`bool std::__profile::operator>= (const multimap<_Key, _Tp, _Compare, _Allocator> &__lhs, const multimap<_Key, _Tp, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>`  
`void std::__profile::swap (multimap<_Key, _Tp, _Compare, _Allocator> &__lhs, multimap<_Key, _Tp, _Compare, _Allocator> &__rhs)`



### 6.199.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/multimap.h](#).

## 6.200 multiseq\_selection.h File Reference

Functions to find elements of a certain global \_\_rank in multiple sorted sequences. Also serves for splitting such sequence sets.

### Classes

- class [\\_\\_gnu\\_parallel::\\_Lexicographic<\\_T1, \\_T2, \\_Compare>](#)  
*Compare \_\_a pair of types lexicographically, ascending.*
- class [\\_\\_gnu\\_parallel::\\_LexicographicReverse<\\_T1, \\_T2, \\_Compare>](#)  
*Compare \_\_a pair of types lexicographically, descending.*

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Defines

- `#define __S(__i)`
- `#define __S(__i)`

### Functions

- `template<typename _RanSeqs, typename _RankType, typename _RankIterator, typename _Compare>`  
`void \_\_gnu\_parallel::multiseq\_partition (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankIterator __begin_offsets, _Compare __comp=std::less< typename std::iterator_traits< typename std::iterator_traits< _RanSeqs >::value_type::first_type >::value_type >())`
- `template<typename _Tp, typename _RanSeqs, typename _RankType, typename _Compare>`  
`_Tp \_\_gnu\_parallel::multiseq\_selection (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankType &__offset, _Compare __comp=std::less< _Tp >())`

### 6.200.1 Detailed Description

Functions to find elements of a certain global `__rank` in multiple sorted sequences. Also serves for splitting such sequence sets. The algorithm description can be found in P. J. Varman, S. D. Scheufler, B. R. Iyer, and G. R. Ricard. Merging Multiple Lists on Hierarchical-Memory Multiprocessors. Journal of Parallel and Distributed Computing, 12(2):171–177, 1991.

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [multiseq\\_selection.h](#).

## 6.201 multiset.h File Reference

### Classes

- class [std::\\_\\_debug::multiset< \\_Key, \\_Compare, \\_Allocator >](#)  
Class [std::multiset](#) with safety/checking/debug instrumentation.

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_debug](#)

### Functions

- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool std::\_\_debug::operator!= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool std::\_\_debug::operator< (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool std::\_\_debug::operator<= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool std::\_\_debug::operator== (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool std::\_\_debug::operator> (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`

- `template<typename _Key , typename _Compare , typename _Allocator >`  
`bool std::__debug::operator>= (const multiset< _Key, _Compare, _Allocator`  
`> &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator >`  
`void std::__debug::swap (multiset< _Key, _Compare, _Allocator > &__x,`  
`multiset< _Key, _Compare, _Allocator > &__y)`

### 6.201.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/multiset.h](#).

## 6.202 multiset.h File Reference

### Classes

- class [std::\\_\\_profile::multiset< \\_Key, \\_Compare, \\_Allocator >](#)  
*Class [std::multiset](#) wrapper with performance instrumentation.*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_profile](#)

### Functions

- `template<typename _Key , typename _Compare , typename _Allocator >`  
`bool std::__profile::operator!= (const multiset< _Key, _Compare, _Allocator`  
`> &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator >`  
`bool std::__profile::operator< (const multiset< _Key, _Compare, _Allocator`  
`> &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator >`  
`bool std::__profile::operator<= (const multiset< _Key, _Compare, _Allocator`  
`> &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator >`  
`bool std::__profile::operator== (const multiset< _Key, _Compare, _Allocator`  
`> &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator >`  
`bool std::__profile::operator> (const multiset< _Key, _Compare, _Allocator`  
`> &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`

- `template<typename _Key , typename _Compare , typename _Allocator >`  
`bool std::__profile::operator>= (const multiset< _Key, _Compare, _Allocator`  
`> &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator >`  
`void std::__profile::swap (multiset< _Key, _Compare, _Allocator > &__x,`  
`multiset< _Key, _Compare, _Allocator > &__y)`

### 6.202.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/multiset.h](#).

## 6.203 multiway\_merge.h File Reference

Implementation of sequential and parallel multiway merge.

### Classes

- `struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __-`  
`sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`  
*Switch for 3-way merging with \_\_sentinels turned off.*
- `struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _-`  
`RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`  
*Switch for 3-way merging with \_\_sentinels turned on.*
- `struct __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __-`  
`sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`  
*Switch for 4-way merging with \_\_sentinels turned off.*
- `struct __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _-`  
`RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`  
*Switch for 4-way merging with \_\_sentinels turned on.*
- `struct __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __-`  
`sentinels, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare`  
`>`  
*Switch for k-way merging with \_\_sentinels turned on.*
- `struct __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, _-`  
`__stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`  
*Switch for k-way merging with \_\_sentinels turned off.*

- class [\\_\\_gnu\\_parallel::\\_GuardedIterator<\\_RAIter, \\_Compare>](#)  
*\_Iterator wrapper supporting an implicit supremum at the end of the sequence, dominating all comparisons.*
- struct [\\_\\_gnu\\_parallel::\\_LoserTreeTraits<\\_Tp>](#)  
*Traits for determining whether the loser tree should use pointers or copies.*
- struct [\\_\\_gnu\\_parallel::\\_SamplingSorter<\\_\\_stable, \\_RAIter, \\_StrictWeakOrdering>](#)  
*Stable sorting functor.*
- struct [\\_\\_gnu\\_parallel::\\_SamplingSorter<false, \\_RAIter, \\_StrictWeakOrdering>](#)  
*Non-\_\_stable sorting functor.*

## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

## Defines

- [#define \\_GLIBCXX\\_PARALLEL\\_DECISION\(\\_\\_a, \\_\\_b, \\_\\_c, \\_\\_d\)](#)
- [#define \\_GLIBCXX\\_PARALLEL\\_LENGTH\(\\_\\_s\)](#)
- [#define \\_GLIBCXX\\_PARALLEL\\_MERGE\\_3\\_CASE\(\\_\\_a, \\_\\_b, \\_\\_c, \\_\\_c0, \\_\\_c1\)](#)
- [#define \\_GLIBCXX\\_PARALLEL\\_MERGE\\_4\\_CASE\(\\_\\_a, \\_\\_b, \\_\\_c, \\_\\_d, \\_\\_c0, \\_\\_c1, \\_\\_c2\)](#)

## Functions

- [template<bool \\_\\_stable, bool \\_\\_sentinels, typename \\_RAIterIterator, typename \\_RAIter3, typename \\_DifferenceTp, typename \\_Compare>](#)  
[\\_\\_gnu\\_parallel::\\_\\_sequential\\_multiway\\_merge](#) ([\\_RAIterIterator](#) \_\_seqs\_begin, [\\_RAIterIterator](#) \_\_seqs\_end, [\\_RAIter3](#) \_\_target, const typename [std::iterator\\_traits<typename std::iterator\\_traits<\\_RAIterIterator>::value\\_type::first\\_type>::value\\_type](#) &\_\_sentinel, [\\_DifferenceTp](#) \_\_length, [\\_Compare](#) \_\_comp)
- [template<typename \\_RAIterPairIterator, typename \\_RAIterOut, typename \\_DifferenceTp, typename \\_Compare>](#)  
[\\_\\_gnu\\_parallel::multiway\\_merge](#) ([\\_RAIterPairIterator](#) \_\_seqs\_begin, [\\_RAIterPairIterator](#) \_\_seqs\_end, [\\_RAIterOut](#) \_\_target, [\\_DifferenceTp](#) \_\_length, [\\_Compare](#) \_\_comp, [\\_\\_gnu\\_parallel::sequential\\_tag](#))

- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut __gnu_parallel::multiway_merge (_RAIterPairIterator __seqs_`  
`begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __`  
`_length, _Compare __comp, __gnu_parallel::exact_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, type-`  
`name _Compare >`  
`_RAIterOut __gnu_parallel::multiway_merge (_RAIterPairIterator __seqs_`  
`begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __`  
`_length, _Compare __comp, __gnu_parallel::sampling_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, type-`  
`name _Compare >`  
`_RAIterOut __gnu_parallel::multiway_merge (_RAIterPairIterator __seqs_`  
`begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __`  
`_length, _Compare __comp, parallel_tag __tag=parallel_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, type-`  
`name _Compare >`  
`_RAIterOut __gnu_parallel::multiway_merge (_RAIterPairIterator __seqs_`  
`begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __`  
`_length, _Compare __comp, default_parallel_tag __tag)`
- `template<template< typename RAI, typename C > class iterator, typename _RAIterIterator ,`  
`typename _RAIter3, typename _DifferenceTp, typename _Compare >`  
`_RAIter3 __gnu_parallel::multiway_merge_3_variant (_RAIterIterator __`  
`seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp`  
`__length, _Compare __comp)`
- `template<template< typename RAI, typename C > class iterator, typename _RAIterIterator ,`  
`typename _RAIter3, typename _DifferenceTp, typename _Compare >`  
`_RAIter3 __gnu_parallel::multiway_merge_4_variant (_RAIterIterator __`  
`seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp`  
`__length, _Compare __comp)`
- `template<bool __stable, typename _RAIterIterator , typename _Compare , typename _`  
`DifferenceType >`  
`void __gnu_parallel::multiway_merge_exact_splitting (_RAIterIterator __`  
`seqs_begin, _RAIterIterator __seqs_end, _DifferenceType __length, _`  
`DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _`  
`DifferenceType, _DifferenceType > > *__pieces)`
- `template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp`  
`, typename _Compare >`  
`_RAIter3 __gnu_parallel::multiway_merge_loser_tree (_RAIterIterator __`  
`seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __`  
`_length, _Compare __comp)`
- `template<typename UnguardedLoserTree , typename _RAIterIterator , typename _RAIter3 ,`  
`typename _DifferenceTp, typename _Compare >`  
`_RAIter3 __gnu_parallel::multiway_merge_loser_tree_sentinel (_`  
`RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __`  
`target, const typename std::iterator_traits< typename std::iterator_traits<`

```

_RAlterIterator >::value_type::first_type >::value_type &__sentinel, _-
DifferenceTp __length, _Compare __comp)
• template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _-
DifferenceTp, typename _Compare >
_RAlter3 __gnu_parallel::multiway_merge_loser_tree_unguarded (_-
RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __-
target, const typename std::iterator_traits< typename std::iterator_traits<
_RAlterIterator >::value_type::first_type >::value_type &__sentinel, _-
DifferenceTp __length, _Compare __comp)
• template<bool __stable, typename _RAIterIterator, typename _Compare, typename _-
DifferenceType >
void __gnu_parallel::multiway_merge_sampling_splitting (_RAIterIterator _-
seqs_begin, _RAIterIterator __seqs_end, _DifferenceType __length, _-
DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _-
DifferenceType, _DifferenceType > > *__pieces)
• template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, type-
name _Compare >
_RAlterOut __gnu_parallel::multiway_merge_sentinels (_RAIterPairIterator
__seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _-
DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))

• template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, type-
name _Compare >
_RAlterOut __gnu_parallel::multiway_merge_sentinels (_RAIterPairIterator _-
seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _-
DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)
• template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, type-
name _Compare >
_RAlterOut __gnu_parallel::multiway_merge_sentinels (_RAIterPairIterator
__seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _-
DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag __tag)
• template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, type-
name _Compare >
_RAlterOut __gnu_parallel::multiway_merge_sentinels (_RAIterPairIterator
__seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _-
DifferenceTp __length, _Compare __comp, sampling_tag __tag)
• template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, type-
name _Compare >
_RAlterOut __gnu_parallel::multiway_merge_sentinels (_RAIterPairIterator
__seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _-
DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)
• template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, type-
name _DifferenceTp, typename _Splitter, typename _Compare >
_RAlter3 __gnu_parallel::parallel_multiway_merge (_RAIterIterator __seqs _-
begin, _RAIterIterator __seqs_end, _RAIter3 __target, _Splitter __splitter, _-
DifferenceTp __length, _Compare __comp, _ThreadIndex __num_threads)

```

- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut __gnu_parallel::stable_multiway_merge (_RAIterPairIterator __`  
`__seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _`  
`DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, type-`  
`name _Compare >`  
`_RAIterOut __gnu_parallel::stable_multiway_merge (_RAIterPairIterator __`  
`__seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _`  
`DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, type-`  
`name _Compare >`  
`_RAIterOut __gnu_parallel::stable_multiway_merge (_RAIterPairIterator __`  
`__seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _`  
`DifferenceTp __length, _Compare __comp, sampling_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, type-`  
`name _Compare >`  
`_RAIterOut __gnu_parallel::stable_multiway_merge (_RAIterPairIterator __`  
`__seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _`  
`DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, type-`  
`name _Compare >`  
`_RAIterOut __gnu_parallel::stable_multiway_merge (_RAIterPairIterator __`  
`__seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _`  
`DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp,`   
`typename _Compare >`  
`_RAIterOut __gnu_parallel::stable_multiway_merge_sentinels (_`  
`RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut`  
`__target, _DifferenceTp __length, _Compare __comp, sampling_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp,`   
`typename _Compare >`  
`_RAIterOut __gnu_parallel::stable_multiway_merge_sentinels (_`  
`RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut`  
`__target, _DifferenceTp __length, _Compare __comp, default_parallel_tag`  
`__tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp,`   
`typename _Compare >`  
`_RAIterOut __gnu_parallel::stable_multiway_merge_sentinels (_`  
`RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut`  
`__target, _DifferenceTp __length, _Compare __comp, parallel_tag __`  
`tag=parallel_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp,`   
`typename _Compare >`



```

_RAlterOut __gnu_parallel::stable_multiway_merge_sentinels (_-
RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut
__target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_-
tag __tag)

```

- template<typename \_RAIterPairIterator , typename \_RAIterOut , typename \_DifferenceTp ,  
typename \_Compare >  
\_RAIterOut \_\_gnu\_parallel::stable\_multiway\_merge\_sentinels ( \_-  
RAIterPairIterator \_\_seqs\_begin, \_RAIterPairIterator \_\_seqs\_end, \_-  
RAIterOut \_\_target, \_DifferenceTp \_\_length, \_Compare \_\_comp, \_\_gnu\_-  
parallel::sequential\_tag)

### 6.203.1 Detailed Description

Implementation of sequential and parallel multiway merge. Explanations on the high-speed merging routines in the appendix of

P. Sanders. Fast priority queues for cached memory. ACM Journal of Experimental Algorithmics, 5, 2000.

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [multiway\\_merge.h](#).

### 6.203.2 Define Documentation

#### 6.203.2.1 #define \_GLIBCXX\_PARALLEL\_LENGTH( \_\_s )

Length of a sequence described by a pair of iterators.

Definition at line 53 of file [multiway\\_merge.h](#).

Referenced by [\\_\\_gnu\\_parallel::\\_\\_sequential\\_multiway\\_merge\(\)](#), [\\_\\_gnu\\_parallel::multiway\\_merge\\_exact\\_splitting\(\)](#), [\\_\\_gnu\\_parallel::multiway\\_merge\\_loser\\_tree\(\)](#), [\\_\\_gnu\\_parallel::multiway\\_merge\\_sampling\\_splitting\(\)](#), and [\\_\\_gnu\\_parallel::parallel\\_multiway\\_merge\(\)](#).

## 6.204 multiway\_mergesort.h File Reference

Parallel multiway merge sort. This file is a GNU parallel extension to the Standard C++ Library.

### Classes

- struct [\\_\\_gnu\\_parallel::Piece< \\_DifferenceTp >](#)

*Subsequence description.*

- struct [\\_\\_gnu\\_parallel::\\_PMWMSSortingData< \\_RAIter >](#)  
*Data accessed by all threads.*
- struct [\\_\\_gnu\\_parallel::\\_SplitConsistently< \\_\\_exact, \\_RAIter, \\_Compare, \\_-SortingPlacesIterator >](#)  
*Split consistently.*
- struct [\\_\\_gnu\\_parallel::\\_SplitConsistently< false, \\_RAIter, \\_Compare, \\_-SortingPlacesIterator >](#)  
*Split by sampling.*
- struct [\\_\\_gnu\\_parallel::\\_SplitConsistently< true, \\_RAIter, \\_Compare, \\_-SortingPlacesIterator >](#)  
*Split by exact splitting.*

## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

## Functions

- template<typename \_RAIter, typename \_DifferenceTp >  
void [\\_\\_gnu\\_parallel::\\_\\_determine\\_samples](#) (\_PMWMSSortingData< \_RAIter > \*\_\_sd, \_DifferenceTp \_\_num\_samples)
- template<bool \_\_stable, bool \_\_exact, typename \_RAIter, typename \_Compare >  
void [\\_\\_gnu\\_parallel::parallel\\_sort\\_mwms](#) (\_RAIter \_\_begin, \_RAIter \_\_end, \_-Compare \_\_comp, \_ThreadIndex \_\_num\_threads)
- template<bool \_\_stable, bool \_\_exact, typename \_RAIter, typename \_Compare >  
void [\\_\\_gnu\\_parallel::parallel\\_sort\\_mwms\\_pu](#) (\_PMWMSSortingData< \_-RAIter > \*\_\_sd, \_Compare &\_\_comp)

### 6.204.1 Detailed Description

Parallel multiway merge sort. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [multiway\\_mergesort.h](#).

## 6.205 mutex File Reference

### Classes

- struct [std::adopt\\_lock\\_t](#)  
*Assume the calling thread has already obtained mutex ownership /// and manage it.*
- struct [std::defer\\_lock\\_t](#)  
*Do not acquire ownership of the mutex.*
- class [std::lock\\_guard< \\_Mutex >](#)  
*Scoped lock idiom.*
- class [std::mutex](#)  
*mutex*
- struct [std::once\\_flag](#)  
*once\_flag*
- class [std::recursive\\_mutex](#)  
*recursive\_mutex*
- class [std::recursive\\_timed\\_mutex](#)  
*recursive\_timed\_mutex*
- class [std::timed\\_mutex](#)  
*timed\_mutex*
- struct [std::try\\_to\\_lock\\_t](#)  
*Try to acquire ownership of the mutex without blocking.*
- class [std::unique\\_lock< \\_Mutex >](#)  
*unique\_lock*

### Namespaces

- namespace [std](#)

## Functions

- mutex & **std::\_\_get\_once\_mutex** ()
- void **std::\_\_once\_proxy** ()
- void **std::\_\_set\_once\_functor\_lock\_ptr** (unique\_lock< mutex > \*)
- template<typename \_Lock >  
unique\_lock< \_Lock > **std::\_\_try\_to\_lock** (\_Lock &\_\_l)
- template<typename \_Callable, typename... \_Args>  
void **std::call\_once** (once\_flag &\_\_once, \_Callable &&\_\_f, \_Args &&...\_\_args)
- template<typename \_L1, typename \_L2, typename... \_L3>  
void **std::lock** (\_L1 &\_\_l1, \_L2 &\_\_l2, \_L3 &...\_\_l3)
- template<typename \_Mutex >  
void **std::swap** (unique\_lock< \_Mutex > &\_\_x, unique\_lock< \_Mutex > &\_\_y)
- template<typename \_Lock1, typename \_Lock2, typename... \_Lock3>  
int **std::try\_lock** (\_Lock1 &\_\_l1, \_Lock2 &\_\_l2, \_Lock3 &...\_\_l3)

## Variables

- function< void()> **std::\_\_once\_functor**

### 6.205.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [mutex](#).

## 6.206 nested\_exception.h File Reference

### Classes

- class [std::nested\\_exception](#)  
*Exception class with exception\_ptr data member.*

### Namespaces

- namespace [std](#)

## Functions

- `template<typename _Ex >`  
`const nested_exception * std::__get_nested_exception (const _Ex &__ex)`
- `template<typename _Ex >`  
`void std::__throw_with_nested (_Ex &&, const nested_exception *__ex) __-`  
`attribute__((__noreturn__))`
- `template<typename _Ex >`  
`void std::__throw_with_nested (_Ex &&, ...) __attribute__((__noreturn__))`
- `void std::rethrow_if_nested (const nested_exception &__ex)`
- `template<typename _Ex >`  
`void std::rethrow_if_nested (const _Ex &__ex)`
- `template<typename _Ex >`  
`void std::throw_with_nested (_Ex __ex)`

### 6.206.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<exception>`.

Definition in file [nested\\_exception.h](#).

## 6.207 new File Reference

### Classes

- class [std::bad\\_alloc](#)  
*Exception possibly thrown by `new`.  
[bad\\_alloc](#) (or classes derived from it) is used to report allocation errors from the throwing forms of `new`.*

### Namespaces

- namespace [std](#)

### Typedefs

- `typedef void(* std::new_handler)()`

## Functions

- new\_handler [std::set\\_new\\_handler](#) (new\_handler) throw ()
- void \* [operator new](#) (std::size\_t) throw (std::bad\_alloc)
- void \* [operator new\[\]](#) (std::size\_t) throw (std::bad\_alloc)
- void [operator delete](#) (void \*) throw ()
- void [operator delete\[\]](#) (void \*) throw ()
- void \* [operator new](#) (std::size\_t, const std::nothrow\_t &) throw ()
- void \* [operator new\[\]](#) (std::size\_t, const std::nothrow\_t &) throw ()
- void [operator delete](#) (void \*, const std::nothrow\_t &) throw ()
- void [operator delete\[\]](#) (void \*, const std::nothrow\_t &) throw ()
- void \* [operator new](#) (std::size\_t, void \*\_\_p) throw ()
- void \* [operator new\[\]](#) (std::size\_t, void \*\_\_p) throw ()
- void [operator delete](#) (void \*, void \*) throw ()
- void [operator delete\[\]](#) (void \*, void \*) throw ()

## Variables

- const nothrow\_t **std::nothrow**

### 6.207.1 Detailed Description

This is a Standard C++ Library header.

The header `new` defines several functions to manage dynamic memory and handling memory allocation errors; see [http://gcc.gnu.org/onlinedocs/libstdc++/18\\_support/howto.html#4](http://gcc.gnu.org/onlinedocs/libstdc++/18_support/howto.html#4) for more.

Definition in file [new](#).

### 6.207.2 Function Documentation

#### 6.207.2.1 void operator delete ( void \* ) throw ()

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

**6.207.2.2 void operator delete ( void \*, const std::nothrow\_t & ) throw ()**

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

**6.207.2.3 void operator delete ( void \*, void \* ) throw () [inline]**

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

Definition at line 107 of file new.

**6.207.2.4 void operator delete[] ( void \* ) throw ()**

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

**6.207.2.5 void operator delete[] ( void \*, const std::nothrow\_t & ) throw ()**

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

**6.207.2.6 void operator delete[] ( void \*, void \* ) throw () [inline]**

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

Definition at line 108 of file new.

**6.207.2.7 void\* operator new ( std::size\_t, void \* \_\_p ) throw () [inline]**

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

Definition at line 103 of file new.



**6.207.2.8 void\* operator new ( std::size\_t, const std::nothrow\_t & ) throw ()**

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

**6.207.2.9 void\* operator new ( std::size\_t ) throw (std::bad\_alloc)**

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

**6.207.2.10 void\* operator new[] ( std::size\_t, const std::nothrow\_t & ) throw ()**

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

**6.207.2.11 void\* operator new[] ( std::size\_t ) throw (std::bad\_alloc)**

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

**6.207.2.12 void\* operator new[] ( std::size\_t, void \* \_\_p ) throw ()  
[inline]**

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

Definition at line 104 of file `new`.

**6.208 new\_allocator.h File Reference****Classes**

- class [`\_\_gnu\_cxx::new\_allocator<\_Tp>`](#)

*An allocator that uses global new, as per [20.4].*

*This is precisely the allocator defined in the C++ Standard.*

- all allocation calls `operator new`
- all deallocation calls `operator delete`.

## Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

## Functions

- `template<typename _Tp >`  
`bool __gnu_cxx::operator!= (const new_allocator< _Tp > &, const new_allocator< _Tp > &)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator== (const new_allocator< _Tp > &, const new_allocator< _Tp > &)`

### 6.208.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [new\\_allocator.h](#).

## 6.209 numeric File Reference

### 6.209.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [numeric](#).

## 6.210 numeric File Reference

## Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

## Functions

- `template<typename _Tp, typename _Integer, typename _MonoidOperation >`  
`_Tp __gnu_cxx::__power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`
- `template<typename _Tp, typename _Integer >`  
`_Tp __gnu_cxx::__power (_Tp __x, _Integer __n)`
- `template<typename _ForwardIter, typename _Tp >`  
`void __gnu_cxx::iota (_ForwardIter __first, _ForwardIter __last, _Tp __value)`

- `template<typename _Tp, typename _Integer, typename _MonoidOperation >`  
`_Tp __gnu_cxx::power (_Tp __x, _Integer __n, _MonoidOperation __monoid_`  
`op)`
- `template<typename _Tp, typename _Integer >`  
`_Tp __gnu_cxx::power (_Tp __x, _Integer __n)`

### 6.210.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [ext/numeric](#).

## 6.211 numeric File Reference

Parallel STL function calls corresponding to [stl\\_numeric.h](#). The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_parallel](#)

### Functions

- `template<typename _Iter, typename _Tp, typename _IteratorTag >`  
`_Tp std::__parallel::__accumulate_switch (_Iter __begin, _Iter __end, _Tp`  
`__init, _IteratorTag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation, typename _IteratorTag >`  
`_Tp std::__parallel::__accumulate_switch (_Iter __begin, _Iter __end, _Tp`  
`__init, _BinaryOperation __binary_op, _IteratorTag)`
- `template<typename __RAIter, typename _Tp, typename _BinaryOperation >`  
`_Tp std::__parallel::__accumulate_switch (__RAIter __begin, __RAIter __`  
`end, _Tp __init, _BinaryOperation __binary_op, random_access_iterator_`  
`tag, __gnu_parallel::__Parallelism __parallelism_tag=__gnu_parallel::parallel_`  
`unbalanced)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation, typename _`  
`IteratorTag1, typename _IteratorTag2 >`  
`_OutputIterator std::__parallel::__adjacent_difference_switch (_Iter __`  
`begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, _`  
`IteratorTag1, _IteratorTag2)`

- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::__parallel::__adjacent_difference_switch (_Iter __`  
`begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op,`  
`random_access_iterator_tag, random_access_iterator_tag, \_\_gnu\_parallel::\_\_`  
`Parallelism __parallelism_tag=\_\_gnu\_parallel::parallel\_balanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _Tp, typename _BinaryFunction1,`  
`typename _BinaryFunction2 >`  
`_Tp std::__parallel::__inner_product_switch (_RAIter1 __first1, _RAIter1`  
`__last1, _RAIter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1,`  
`_BinaryFunction2 __binary_op2, random_access_iterator_tag, random_`  
`access_iterator_tag, \_\_gnu\_parallel::\_\_Parallelism __parallelism_tag=\_\_gnu\_`  
`parallel::parallel\_unbalanced)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, type-`  
`name _BinaryFunction2, typename _IteratorTag1, typename _IteratorTag2 >`  
`_Tp std::__parallel::__inner_product_switch (_Iter1 __first1, _Iter1 __`  
`__last1, _Iter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _`  
`BinaryFunction2 __binary_op2, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation, typename _`  
`IteratorTag1, typename _IteratorTag2 >`  
`_OutputIterator std::__parallel::__partial_sum_switch (_Iter __begin, _Iter`  
`__end, _OutputIterator __result, _BinaryOperation __bin_op, _IteratorTag1, _`  
`IteratorTag2)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::__parallel::__partial_sum_switch (_Iter __begin, _Iter`  
`__end, _OutputIterator __result, _BinaryOperation __bin_op, random_access_`  
`iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation >`  
`_Tp std::__parallel::__accumulate (_Iter __begin, _Iter __end, _Tp __init, _`  
`BinaryOperation __binary_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation >`  
`_Tp std::__parallel::__accumulate (_Iter __begin, _Iter __end, _Tp __init, _`  
`BinaryOperation __binary_op, \_\_gnu\_parallel::\_\_Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation >`  
`_Tp std::__parallel::__accumulate (_Iter __begin, _Iter __end, _Tp __init, _`  
`BinaryOperation __binary_op)`
- `template<typename _Iter, typename _Tp >`  
`_Tp std::__parallel::__accumulate (_Iter __begin, _Iter __end, _Tp __init, \_\_`  
`gnu\_parallel::\_\_Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp >`  
`_Tp std::__parallel::__accumulate (_Iter __begin, _Iter __end, _Tp __init, \_\_`  
`gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp >`  
`_Tp std::__parallel::__accumulate (_Iter __begin, _Iter __end, _Tp __init)`

- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __binary_op, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`  
`_Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`  
`_Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`  
`_Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`  
`_Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`  
`_Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`  
`_Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::__parallel::partial_sum (_Iter __begin, _Iter __end, _-`  
`OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator std::__parallel::partial_sum (_Iter __begin, _Iter __end, _-`  
`OutputIterator __result)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator std::__parallel::partial_sum (_Iter __begin, _Iter __end, _-`  
`OutputIterator __result, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::__parallel::partial_sum (_Iter __begin, _Iter _-`  
`end, _OutputIterator __result, _BinaryOperation __bin_op, \_\_gnu-`  
`parallel::sequential\_tag)`

### 6.211.1 Detailed Description

Parallel STL function calls corresponding to [stl\\_numeric.h](#). The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [parallel/numeric](#).

## 6.212 `numeric_traits.h` File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Defines

- `#define \_\_glibcxx\_digits(_Tp)`
- `#define \_\_glibcxx\_digits10(_Tp)`
- `#define \_\_glibcxx\_floating(_Tp, _Fval, _Dval, _LDval)`
- `#define \_\_glibcxx\_max(_Tp)`
- `#define \_\_glibcxx\_max\_digits10(_Tp)`
- `#define \_\_glibcxx\_max\_exponent10(_Tp)`
- `#define \_\_glibcxx\_min(_Tp)`
- `#define \_\_glibcxx\_signed(_Tp)`

### 6.212.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [numeric\\_traits.h](#).

## 6.213 numericfwd.h File Reference

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_parallel](#)

### Functions

- `template<typename _Iter, typename _Tp, typename _Tag >  
_Tp std::__parallel::__accumulate_switch (_Iter, _Iter, _Tp, _Tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper, typename _Tag >  
_Tp std::__parallel::__accumulate_switch (_Iter, _Iter, _Tp, _BinaryOper,  
_Tag)`
- `template<typename _RAIter, typename _Tp, typename _BinaryOper >  
_Tp std::__parallel::__accumulate_switch (_RAIter, _RAIter, _Tp, _-  
BinaryOper, random_access_iterator_tag, \_\_gnu\_parallel::\_\_Parallelism __-  
parallelism=\_\_gnu\_parallel::parallel\_unbalanced)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper, typename _Tag1, typename  
_Tag2 >  
_OIter std::__parallel::__adjacent_difference_switch (_Iter, _Iter, _OIter, _-  
BinaryOper, _Tag1, _Tag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >  
_OIter std::__parallel::__adjacent_difference_switch (_Iter, _Iter, _OIter, _-  
BinaryOper, random_access_iterator_tag, random_access_iterator_tag, \_\_gnu\_-  
parallel::\_\_Parallelism __parallelism=\_\_gnu\_parallel::parallel\_unbalanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _Tp, typename BinaryFunction1 ,  
typename BinaryFunction2 >  
_Tp std::__parallel::__inner_product_switch (_RAIter1, _RAIter1, _-  
RAIter2, _Tp, BinaryFunction1, BinaryFunction2, random_access_iterator_-  
tag, random_access_iterator_tag, \_\_gnu\_parallel::\_\_Parallelism=\_\_gnu\_-  
parallel::parallel\_unbalanced)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, type-  
name _BinaryFunction2, typename _Tag1, typename _Tag2 >  
_Tp std::__parallel::__inner_product_switch (_Iter1, _Iter1, _Iter2, _Tp,  
_BinaryFunction1, _BinaryFunction2, _Tag1, _Tag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper, typename _Tag1, typename  
_Tag2 >`



- `_OIter std::__parallel::__partial_sum_switch` (`_IIter`, `_IIter`, `_OIter`, `_BinaryOper`, `_Tag1`, `_Tag2`)
- `template<typename _IIter, typename _OIter, typename _BinaryOper >`  
`_OIter std::__parallel::__partial_sum_switch` (`_IIter`, `_IIter`, `_OIter`, `_BinaryOper`, `random_access_iterator_tag`, `random_access_iterator_tag`)
  - `template<typename _IIter, typename _Tp >`  
`_Tp std::__parallel::__accumulate` (`_IIter __begin`, `_IIter __end`, `_Tp __init`, `__gnu_parallel::__sequential_tag`)
  - `template<typename _IIter, typename _Tp, typename _BinaryOper >`  
`_Tp std::__parallel::__accumulate` (`_IIter`, `_IIter`, `_Tp`, `_BinaryOper`, `__gnu_parallel::__Parallelism`)
  - `template<typename _IIter, typename _Tp >`  
`_Tp std::__parallel::__accumulate` (`_IIter __begin`, `_IIter __end`, `_Tp __init`)
  - `template<typename _IIter, typename _Tp, typename _BinaryOper >`  
`_Tp std::__parallel::__accumulate` (`_IIter`, `_IIter`, `_Tp`, `_BinaryOper`)
  - `template<typename _IIter, typename _Tp >`  
`_Tp std::__parallel::__accumulate` (`_IIter __begin`, `_IIter __end`, `_Tp __init`, `__gnu_parallel::__Parallelism __parallelism_tag`)
  - `template<typename _IIter, typename _Tp, typename _BinaryOper >`  
`_Tp std::__parallel::__accumulate` (`_IIter`, `_IIter`, `_Tp`, `_BinaryOper`, `__gnu_parallel::__sequential_tag`)
  - `template<typename _IIter, typename _OIter, typename _BinaryOper >`  
`_OIter std::__parallel::__adjacent_difference` (`_IIter`, `_IIter`, `_OIter`, `_BinaryOper`, `__gnu_parallel::__Parallelism`)
  - `template<typename _IIter, typename _OIter >`  
`_OIter std::__parallel::__adjacent_difference` (`_IIter`, `_IIter`, `_OIter`)
  - `template<typename _IIter, typename _OIter, typename _BinaryOper >`  
`_OIter std::__parallel::__adjacent_difference` (`_IIter`, `_IIter`, `_OIter`, `_BinaryOper`)
  - `template<typename _IIter, typename _OIter >`  
`_OIter std::__parallel::__adjacent_difference` (`_IIter`, `_IIter`, `_OIter`, `__gnu_parallel::__sequential_tag`)
  - `template<typename _IIter, typename _OIter, typename _BinaryOper >`  
`_OIter std::__parallel::__adjacent_difference` (`_IIter`, `_IIter`, `_OIter`, `_BinaryOper`, `__gnu_parallel::__sequential_tag`)
  - `template<typename _IIter, typename _OIter >`  
`_OIter std::__parallel::__adjacent_difference` (`_IIter`, `_IIter`, `_OIter`, `__gnu_parallel::__Parallelism`)
  - `template<typename _IIter1, typename _IIter2, typename _Tp >`  
`_Tp std::__parallel::__inner_product` (`_IIter1 __first1`, `_IIter1 __last1`, `_IIter2 __first2`, `_Tp __init`, `__gnu_parallel::__Parallelism __parallelism_tag`)
  - `template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`  
`_Tp std::__parallel::__inner_product` (`_IIter1 __first1`, `_IIter1 __last1`, `_IIter2`

```
__first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __-
binary_op2, __gnu_parallel::sequential_tag)
```

- `template<typename _Iter1, typename _Iter2, typename _Tp, typename BinaryFunction1, type-  
name BinaryFunction2 >`  
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, BinaryFunc-  
tion1, BinaryFunction2, \_\_gnu\_parallel::\_Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, type-  
name _BinaryFunction2 >`  
`_Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2  
__first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __-  
binary_op2)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`  
`_Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 _-  
__first2, _Tp __init)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`  
`_Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 _-  
__first2, _Tp __init, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`  
`_OIter std::__parallel::partial_sum (_Iter, _Iter, _OIter, _BinaryOper)`
- `template<typename _Iter, typename _OIter >`  
`_OIter std::__parallel::partial_sum (_Iter, _Iter, _OIter __result)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`  
`_OIter std::__parallel::partial_sum (_Iter, _Iter, _OIter, _BinaryOper, \_\_-  
gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter >`  
`_OIter std::__parallel::partial_sum (_Iter, _Iter, _OIter, \_\_gnu\_-  
parallel::sequential\_tag)`

### 6.213.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [numericfwd.h](#).

## 6.214 omp\_loop.h File Reference

Parallelization of embarrassingly parallel execution by means of an OpenMP for loop.  
This file is a GNU parallel extension to the Standard C++ Library.

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

## Functions

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>  
>  
_Op __gnu_parallel::__for_each_template_random_access_omp_loop (_RAIter  
__begin, _RAIter __end, _Op __o, _Fu &__f, _Red __r, _Result __base, _-  
Result &__output, typename std::iterator_traits< _RAIter >::difference_type __-  
__bound)`

### 6.214.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of an OpenMP for loop. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [omp\\_loop.h](#).

## 6.215 omp\_loop\_static.h File Reference

Parallelization of embarrassingly parallel execution by means of an OpenMP for loop with static scheduling. This file is a GNU parallel extension to the Standard C++ Library.

## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

## Functions

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>  
>  
_Op __gnu_parallel::__for_each_template_random_access_omp_loop_static  
(_RAIter __begin, _RAIter __end, _Op __o, _Fu &__f, _Red __r, _-  
Result __base, _Result &__output, typename std::iterator_traits< _RAIter  
>::difference_type __bound)`

### 6.215.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of an OpenMP for loop with static scheduling. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [omp\\_loop\\_static.h](#).

## 6.216 os\_defines.h File Reference

### Defines

- `#define __NO_CTYPE`

### 6.216.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

Definition in file [os\\_defines.h](#).

## 6.217 ostream File Reference

### Classes

- class [std::basic\\_ostream<\\_CharT, \\_Traits>](#)  
*Controlling output.*  
*This is the base class for all output streams. It provides text formatting of all builtin types, and communicates with any class derived from [basic\\_streambuf](#) to do the actual output.*
- class [std::basic\\_ostream<\\_CharT, \\_Traits>::sentry](#)  
*Performs setup work for output streams.*

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _CharT, typename _Traits>`  
`basic_ostream<_CharT, _Traits> & std::endl (basic_ostream<_CharT, _Traits> & __os)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream<_CharT, _Traits> & std::ends (basic_ostream<_CharT, _Traits> & __os)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream<_CharT, _Traits> & std::flush (basic_ostream<_CharT, _Traits> & __os)`

- `template<typename _CharT, typename _Traits, typename _Tp>`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-`  
`CharT, _Traits > &&__os, const _Tp &__x)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-`  
`CharT, _Traits > &__out, _CharT __c)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-`  
`CharT, _Traits > &__out, char __c)`
- `template<class _Traits>`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char,`  
`_Traits > &__out, char __c)`
- `template<class _Traits>`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char,`  
`_Traits > &__out, signed char __c)`
- `template<class _Traits>`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char,`  
`_Traits > &__out, unsigned char __c)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-`  
`CharT, _Traits > &__out, const _CharT *__s)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-`  
`CharT, _Traits > &__out, const char *__s)`
- `template<class _Traits>`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char,`  
`_Traits > &__out, const char *__s)`
- `template<class _Traits>`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char,`  
`_Traits > &__out, const signed char *__s)`
- `template<class _Traits>`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char,`  
`_Traits > &__out, const unsigned char *__s)`

### 6.217.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [ostream](#).

## 6.218 ostream.tcc File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _CharT, typename _Traits >  
basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__out, const char *__s)`

#### 6.218.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ostream>`.

Definition in file [ostream.tcc](#).

## 6.219 ostream\_insert.h File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _CharT, typename _Traits >  
void std::__ostream_fill (basic_ostream< _CharT, _Traits > &__out, streamsize __n)`
- `template wostream & std::__ostream_insert (wostream &, const wchar_t *, streamsize)`
- `template ostream & std::__ostream_insert (ostream &, const char *, streamsize)`
- `template<typename _CharT, typename _Traits >  
basic_ostream< _CharT, _Traits > & std::__ostream_insert (basic_ostream< _CharT, _Traits > &__out, const _CharT *__s, streamsize __n)`
- `template<typename _CharT, typename _Traits >  
void std::__ostream_write (basic_ostream< _CharT, _Traits > &__out, const _CharT *__s, streamsize __n)`

### 6.219.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ostream>`.

Definition in file [ostream\\_insert.h](#).

## 6.220 `par_loop.h` File Reference

Parallelization of embarrassingly parallel execution by means of equal splitting. This file is a GNU parallel extension to the Standard C++ Library.

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>  
> _Op __gnu_parallel::__for_each_template_random_access_ed (_RAIter __-  
begin, _RAIter __end, _Op __o, _Fu &__f, _Red __r, _Result __base, _-  
Result &__output, typename std::iterator_traits<_RAIter>::difference_type __-  
_bound)`

### 6.220.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of equal splitting. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [par\\_loop.h](#).

## 6.221 `parallel.h` File Reference

End-user include file. Provides advanced settings and tuning options. This file is a GNU parallel extension to the Standard C++ Library.

### 6.221.1 Detailed Description

End-user include file. Provides advanced settings and tuning options. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [parallel.h](#).

## 6.222 `partial_sum.h` File Reference

Parallel implementation of `std::partial_sum()`, i.e. prefix sums. This file is a GNU parallel extension to the Standard C++ Library.

### Namespaces

- namespace `__gnu_parallel`

### Functions

- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation > _OutputIterator __gnu_parallel::__parallel_partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation > _OutputIterator __gnu_parallel::__parallel_partial_sum_basecase (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, typename std::iterator_traits<_Iter>::value_type __value)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation > _OutputIterator __gnu_parallel::__parallel_partial_sum_linear (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, typename std::iterator_traits<_Iter>::difference_type __n)`

#### 6.222.1 Detailed Description

Parallel implementation of `std::partial_sum()`, i.e. prefix sums. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file `partial_sum.h`.

## 6.223 `partition.h` File Reference

Parallel implementation of `std::partition()`, `std::nth_element()`, and `std::partial_sort()`. This file is a GNU parallel extension to the Standard C++ Library.

### Namespaces

- namespace `__gnu_parallel`

### Defines

- `#define _GLIBCXX_VOLATILE`



## Functions

- `template<typename _RAIter, typename _Compare >`  
`void __gnu_parallel::__parallel_nth_element (_RAIter __begin, _RAIter __nth,`  
`_RAIter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare >`  
`void __gnu_parallel::__parallel_partial_sort (_RAIter __begin, _RAIter __-`  
`middle, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Predicate >`  
`std::iterator_traits< _RAIter >::difference_type __gnu_parallel::__parallel_`  
`partition (_RAIter __begin, _RAIter __end, _Predicate __pred, _ThreadIndex`  
`__num_threads)`

### 6.223.1 Detailed Description

Parallel implementation of `std::partition()`, `std::nth_element()`, and `std::partial_sort()`. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [partition.h](#).

### 6.223.2 Define Documentation

#### 6.223.2.1 #define \_GLIBCXX\_VOLATILE

Decide whether to declare certain variables volatile.

Definition at line 43 of file [partition.h](#).

Referenced by `__gnu_parallel::__parallel_partition()`.

## 6.224 pod\_char\_traits.h File Reference

### Classes

- `struct __gnu_cxx::character< V, I, S >`  
*A POD class that serves as a character abstraction class.*
- `struct std::char_traits< __gnu_cxx::character< V, I, S > >`  
*char\_traits<\_\_gnu\_cxx::character> specialization.*

## Namespaces

- namespace `__gnu_cxx`
- namespace `std`

## Functions

- `template<typename V, typename I, typename S>`  
`bool __gnu_cxx::operator< (const character< V, I, S > &lhs, const character< V, I, S > &rhs)`
- `template<typename V, typename I, typename S>`  
`bool __gnu_cxx::operator== (const character< V, I, S > &lhs, const character< V, I, S > &rhs)`

### 6.224.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file `pod_char_traits.h`.

## 6.225 `pointer.h` File Reference

### Classes

- struct `__gnu_cxx::_Invalid_type`
- class `__gnu_cxx::_Pointer_adapter< _Storage_policy >`
- class `__gnu_cxx::_Relative_pointer_impl< _Tp >`  
*A storage policy for use with `_Pointer_adapter<>` which stores the pointer's address as an offset value which is relative to its own address.*
- class `__gnu_cxx::_Relative_pointer_impl< const _Tp >`
- class `__gnu_cxx::_Std_pointer_impl< _Tp >`  
*A storage policy for use with `_Pointer_adapter<>` which yields a standard pointer.*
- struct `__gnu_cxx::_Unqualified_type< _Tp >`

### Namespaces

- namespace `__gnu_cxx`

## Defines

- `#define _CXX_POINTER_ARITH_OPERATOR_SET(INT_TYPE)`
- `#define _GCC_CXX_POINTER_COMPARISON_OPERATION_SET(OPERATOR)`

## Functions

- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator!= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2`  
`__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator!= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 >`  
`&__rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator!= (const _Pointer_adapter< _Tp > &__lhs, int __`  
`rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator!= (int __lhs, const _Pointer_adapter< _Tp > &__`  
`rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator!= (const _Pointer_adapter< _Tp1 > &__lhs, const`  
`_Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator!= (const _Pointer_adapter< _Tp > &__lhs, const`  
`_Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator< (_Tp1 __lhs, const _Pointer_adapter< _Tp2 >`  
`&__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator< (const _Pointer_adapter< _Tp1 > &__lhs, const`  
`_Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator< (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2`  
`__rhs)`
- `template<typename _CharT, typename _Traits, typename _StoreT >`  
`std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<<`  
`(std::basic_ostream< _CharT, _Traits > &__os, const _Pointer_adapter<`  
`_StoreT > &__p)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator<= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2`  
`__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator<= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 >`  
`&__rhs)`

- `template<typename _Tp >`  
`bool __gnu_cxx::operator<= (const _Pointer_adapter< _Tp > &__lhs, const`  
`_Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator<= (const _Pointer_adapter< _Tp1 > &__lhs, const`  
`_Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp > &__lhs, const _`  
`Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator== (_Tp1 __lhs, const _Pointer_adapter< _Tp2 >`  
`&__rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp > &__lhs, int __`  
`rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator== (int __lhs, const _Pointer_adapter< _Tp > &__`  
`rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2`  
`__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp1 > &__lhs, const`  
`_Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator> (const _Pointer_adapter< _Tp1 > &__lhs, const`  
`_Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator> (const _Pointer_adapter< _Tp > &__lhs, const _`  
`Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator> (_Tp1 __lhs, const _Pointer_adapter< _Tp2 >`  
`&__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator> (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2`  
`__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator>= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2`  
`__rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator>= (const _Pointer_adapter< _Tp > &__lhs, const`  
`_Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator>= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 >`  
`&__rhs)`

- template<typename \_Tp1 , typename \_Tp2 >  
bool **\_\_gnu\_cxx::operator>=** (const \_Pointer\_adapter< \_Tp1 > &\_\_lhs, const  
\_Pointer\_adapter< \_Tp2 > &\_\_rhs)

### 6.225.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

#### Author

Bob Walters

Provides reusable \_Pointer\_adapter for assisting in the development of custom pointer types that can be used with the standard containers via the allocator::pointer and allocator::const\_pointer typedefs.

Definition in file [pointer.h](#).

## 6.226 pool\_allocator.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::\\_\\_pool\\_alloc< \\_Tp >](#)  
*Allocator using a memory pool with a single lock.*
- class [\\_\\_gnu\\_cxx::\\_\\_pool\\_alloc\\_base](#)  
*Base class for [\\_\\_pool\\_alloc](#).*

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Functions

- template<typename \_Tp >  
bool **\_\_gnu\_cxx::operator!=** (const \_\_pool\_alloc< \_Tp > &, const \_\_pool\_  
alloc< \_Tp > &)
- template<typename \_Tp >  
bool **\_\_gnu\_cxx::operator==** (const \_\_pool\_alloc< \_Tp > &, const \_\_pool\_  
alloc< \_Tp > &)

### 6.226.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [pool\\_allocator.h](#).

## 6.227 postypes.h File Reference

### Classes

- class [std::fpos<\\_StateT>](#)  
*Class representing stream positions.*

### Namespaces

- namespace [std](#)

### Typedefs

- typedef long long [std::streamoff](#)
- typedef fpos< mbstate\_t > [std::streampos](#)
- typedef ptrdiff\_t [std::streamsize](#)
- typedef fpos< mbstate\_t > [std::u16streampos](#)
- typedef fpos< mbstate\_t > [std::u32streampos](#)
- typedef fpos< mbstate\_t > [std::wstreampos](#)

### Functions

- template<typename \_StateT >  
bool **std::operator!=** (const fpos< \_StateT > &\_\_lhs, const fpos< \_StateT > &\_\_rhs)
- template<typename \_StateT >  
bool [std::operator==](#) (const fpos< \_StateT > &\_\_lhs, const fpos< \_StateT > &\_\_rhs)

### 6.227.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

Definition in file [postypes.h](#).

## 6.228 `priority_queue.hpp` File Reference

### Namespaces

- namespace [`\_\_gnu\_pbds`](#)

#### 6.228.1 Detailed Description

Contains `priority_queues`.

Definition in file [priority\\_queue.hpp](#).

## 6.229 `priority_queue_base_dispatch.hpp` File Reference

### Namespaces

- namespace [`\_\_gnu\_pbds`](#)

#### 6.229.1 Detailed Description

Contains an pqiative container dispatching base.

Definition in file [priority\\_queue\\_base\\_dispatch.hpp](#).

## 6.230 `profiler.h` File Reference

Interface of the profiling runtime library.

### Classes

- struct [`\_\_gnu\_profile::\_\_reentrance\_guard`](#)  
*Reentrance guard.*

### Namespaces

- namespace [`\_\_gnu\_profile`](#)

### Defines

- `#define __profcxx_hashtable_construct(__x...)`
- `#define __profcxx_hashtable_construct2(__x...)`

- #define **\_\_profcxx\_hashtable\_destruct**(\_\_x...)
- #define **\_\_profcxx\_hashtable\_destruct2**(\_\_x...)
- #define **\_\_profcxx\_hashtable\_resize**(\_\_x...)
- #define **\_\_profcxx\_is\_invalid**()
- #define **\_\_profcxx\_is\_off**()
- #define **\_\_profcxx\_is\_on**()
- #define **\_\_profcxx\_list\_construct**(\_\_x...)
- #define **\_\_profcxx\_list\_construct2**(\_\_x...)
- #define **\_\_profcxx\_list\_destruct**(\_\_x...)
- #define **\_\_profcxx\_list\_destruct2**(\_\_x...)
- #define **\_\_profcxx\_list\_insert**(\_\_x...)
- #define **\_\_profcxx\_list\_invalid\_operator**(\_\_x...)
- #define **\_\_profcxx\_list\_iterate**(\_\_x...)
- #define **\_\_profcxx\_list\_operation**(\_\_x...)
- #define **\_\_profcxx\_list\_rewind**(\_\_x...)
- #define **\_\_profcxx\_map\_to\_unordered\_map\_construct**(\_\_x...)
- #define **\_\_profcxx\_map\_to\_unordered\_map\_destruct**(\_\_x...)
- #define **\_\_profcxx\_map\_to\_unordered\_map\_erase**(\_\_x...)
- #define **\_\_profcxx\_map\_to\_unordered\_map\_find**(\_\_x...)
- #define **\_\_profcxx\_map\_to\_unordered\_map\_insert**(\_\_x...)
- #define **\_\_profcxx\_map\_to\_unordered\_map\_invalidate**(\_\_x...)
- #define **\_\_profcxx\_map\_to\_unordered\_map\_iterate**(\_\_x...)
- #define **\_\_profcxx\_report**()
- #define **\_\_profcxx\_turn\_off**()
- #define **\_\_profcxx\_turn\_on**()
- #define **\_\_profcxx\_vector\_construct**(\_\_x...)
- #define **\_\_profcxx\_vector\_construct2**(\_\_x...)
- #define **\_\_profcxx\_vector\_destruct**(\_\_x...)
- #define **\_\_profcxx\_vector\_destruct2**(\_\_x...)
- #define **\_\_profcxx\_vector\_find**(\_\_x...)
- #define **\_\_profcxx\_vector\_insert**(\_\_x...)
- #define **\_\_profcxx\_vector\_invalid\_operator**(\_\_x...)
- #define **\_\_profcxx\_vector\_iterate**(\_\_x...)
- #define **\_\_profcxx\_vector\_resize**(\_\_x...)
- #define **\_\_profcxx\_vector\_resize2**(\_\_x...)
- #define **\_GLIBCXX\_PROFILE\_DATA**(\_\_name)
- #define **\_GLIBCXX\_PROFILE\_DEFINE\_DATA**(\_\_type, \_\_name, \_\_initial\_value...)
- #define **\_GLIBCXX\_PROFILE\_DEFINE\_UNINIT\_DATA**(\_\_type, \_\_name)
- #define **\_GLIBCXX\_PROFILE\_MAX\_STACK\_DEPTH**
- #define **\_GLIBCXX\_PROFILE\_MAX\_STACK\_DEPTH\_ENV\_VAR**
- #define **\_GLIBCXX\_PROFILE\_MAX\_WARN\_COUNT**
- #define **\_GLIBCXX\_PROFILE\_MAX\_WARN\_COUNT\_ENV\_VAR**



- `#define _GLIBCXX_PROFILE_MEM_PER_DIAGNOSTIC`
- `#define _GLIBCXX_PROFILE_MEM_PER_DIAGNOSTIC_ENV_VAR`
- `#define _GLIBCXX_PROFILE_REENTRANCE_GUARD(__x...)`
- `#define _GLIBCXX_PROFILE_TRACE_ENV_VAR`
- `#define _GLIBCXX_PROFILE_TRACE_PATH_ROOT`

## Functions

- `bool __gnu_profile::__is_invalid ()`
- `bool __gnu_profile::__is_off ()`
- `bool __gnu_profile::__is_on ()`
- `void __gnu_profile::__report (void)`
- `void __gnu_profile::__trace_hash_func_construct (const void *)`
- `void __gnu_profile::__trace_hash_func_destruct (const void *, std::size_t, std::size_t, std::size_t)`
- `void __gnu_profile::__trace_hashtable_size_construct (const void *, std::size_t)`
- `void __gnu_profile::__trace_hashtable_size_destruct (const void *, std::size_t, std::size_t)`
- `void __gnu_profile::__trace_hashtable_size_resize (const void *, std::size_t, std::size_t)`
- `void __gnu_profile::__trace_list_to_set_construct (const void *)`
- `void __gnu_profile::__trace_list_to_set_destruct (const void *)`
- `void __gnu_profile::__trace_list_to_set_find (const void *, std::size_t)`
- `void __gnu_profile::__trace_list_to_set_insert (const void *, std::size_t, std::size_t)`
- `void __gnu_profile::__trace_list_to_set_invalid_operator (const void *)`
- `void __gnu_profile::__trace_list_to_set_iterate (const void *, std::size_t)`
- `void __gnu_profile::__trace_list_to_slist_construct (const void *)`
- `void __gnu_profile::__trace_list_to_slist_destruct (const void *)`
- `void __gnu_profile::__trace_list_to_slist_operation (const void *)`
- `void __gnu_profile::__trace_list_to_slist_rewind (const void *)`
- `void __gnu_profile::__trace_list_to_vector_construct (const void *)`
- `void __gnu_profile::__trace_list_to_vector_destruct (const void *)`
- `void __gnu_profile::__trace_list_to_vector_insert (const void *, std::size_t, std::size_t)`
- `void __gnu_profile::__trace_list_to_vector_invalid_operator (const void *)`
- `void __gnu_profile::__trace_list_to_vector_iterate (const void *, std::size_t)`
- `void __gnu_profile::__trace_list_to_vector_resize (const void *, std::size_t, std::size_t)`
- `void __gnu_profile::__trace_map_to_unordered_map_construct (const void *)`

- void `__gnu_profile::__trace_map_to_unordered_map_destruct` (const void \*)
- void `__gnu_profile::__trace_map_to_unordered_map_erase` (const void \*, std::size\_t, std::size\_t)
- void `__gnu_profile::__trace_map_to_unordered_map_find` (const void \*, std::size\_t)
- void `__gnu_profile::__trace_map_to_unordered_map_insert` (const void \*, std::size\_t, std::size\_t)
- void `__gnu_profile::__trace_map_to_unordered_map_invalidate` (const void \*)
- void `__gnu_profile::__trace_map_to_unordered_map_iterate` (const void \*, std::size\_t)
- void `__gnu_profile::__trace_vector_size_construct` (const void \*, std::size\_t)
- void `__gnu_profile::__trace_vector_size_destruct` (const void \*, std::size\_t, std::size\_t)
- void `__gnu_profile::__trace_vector_size_resize` (const void \*, std::size\_t, std::size\_t)
- void `__gnu_profile::__trace_vector_to_list_construct` (const void \*)
- void `__gnu_profile::__trace_vector_to_list_destruct` (const void \*)
- void `__gnu_profile::__trace_vector_to_list_find` (const void \*, std::size\_t)
- void `__gnu_profile::__trace_vector_to_list_insert` (const void \*, std::size\_t, std::size\_t)
- void `__gnu_profile::__trace_vector_to_list_invalid_operator` (const void \*)
- void `__gnu_profile::__trace_vector_to_list_iterate` (const void \*, std::size\_t)
- void `__gnu_profile::__trace_vector_to_list_resize` (const void \*, std::size\_t, std::size\_t)
- bool `__gnu_profile::__turn_off` ()
- bool `__gnu_profile::__turn_on` ()

### 6.230.1 Detailed Description

Interface of the profiling runtime library.

Definition in file [profiler.h](#).

## 6.231 profiler\_algos.h File Reference

Algorithms used by the profile extension.

### Namespaces

- namespace [\\_\\_gnu\\_profile](#)

## Functions

- `template<typename _InputIterator, typename _Function >  
_Function __gnu_profile::__for_each (_InputIterator __first, _InputIterator __-  
last, _Function __f)`
- `template<typename _Container >  
void __gnu_profile::__insert_top_n (_Container &__output, const typename  
_Container::value_type &__value, typename _Container::size_type __n)`
- `template<typename _ForwardIterator, typename _Tp >  
_ForwardIterator __gnu_profile::__remove (_ForwardIterator __first, _-  
ForwardIterator __last, const _Tp &__value)`
- `template<typename _Container >  
void __gnu_profile::__top_n (const _Container &__input, _Container &__-  
output, typename _Container::size_type __n)`

### 6.231.1 Detailed Description

Algorithms used by the profile extension. This file is needed to avoid including `<algorithm>` or `<bits/stl_algo.h>`. Including those files would result in recursive includes. These implementations are oversimplified. In general, efficiency may be sacrificed to minimize maintenance overhead.

Definition in file [profiler\\_algos.h](#).

## 6.232 profiler\_container\_size.h File Reference

Diagnostics for container sizes.

### Classes

- class [\\_\\_gnu\\_profile::\\_\\_container\\_size\\_info](#)  
*A container size instrumentation line in the object table.*
- class [\\_\\_gnu\\_profile::\\_\\_container\\_size\\_stack\\_info](#)  
*A container size instrumentation line in the stack table.*
- class [\\_\\_gnu\\_profile::\\_\\_trace\\_container\\_size](#)  
*Container size instrumentation trace producer.*

### Namespaces

- namespace [\\_\\_gnu\\_profile](#)

### 6.232.1 Detailed Description

Diagnostics for container sizes.

Definition in file [profiler\\_container\\_size.h](#).

## 6.233 profiler\_hash\_func.h File Reference

Data structures to represent profiling traces.

### Classes

- class [\\_\\_gnu\\_profile::\\_\\_hashfunc\\_info](#)  
*A hash performance instrumentation line in the object table.*
- class [\\_\\_gnu\\_profile::\\_\\_hashfunc\\_stack\\_info](#)  
*A hash performance instrumentation line in the stack table.*
- class [\\_\\_gnu\\_profile::\\_\\_trace\\_hash\\_func](#)  
*Hash performance instrumentation producer.*

### Namespaces

- namespace [\\_\\_gnu\\_profile](#)

### Functions

- void [\\_\\_gnu\\_profile::\\_\\_trace\\_hash\\_func\\_construct](#) (const void \*)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_hash\\_func\\_destruct](#) (const void \*, std::size\_t, std::size\_t)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_hash\\_func\\_init](#) ()
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_hash\\_func\\_report](#) (FILE \*\_\_f, \_\_warning\_vector\_t &\_\_warnings)

### 6.233.1 Detailed Description

Data structures to represent profiling traces.

Definition in file [profiler\\_hash\\_func.h](#).

## 6.234 profiler\_hashtable\_size.h File Reference

Collection of hashtable size traces.

### Classes

- class [\\_\\_gnu\\_profile::\\_\\_trace\\_hashtable\\_size](#)  
*Hashtable size instrumentation trace producer.*

### Namespaces

- namespace [\\_\\_gnu\\_profile](#)

### Functions

- void [\\_\\_gnu\\_profile::\\_\\_trace\\_hashtable\\_size\\_construct](#) (const void \*, std::size\_t)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_hashtable\\_size\\_destruct](#) (const void \*, std::size\_t, std::size\_t)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_hashtable\\_size\\_init](#) ()
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_hashtable\\_size\\_report](#) (FILE \*\_\_f, \_\_warning\_vector\_t &\_\_warnings)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_hashtable\\_size\\_resize](#) (const void \*, std::size\_t, std::size\_t)

#### 6.234.1 Detailed Description

Collection of hashtable size traces.

Definition in file [profiler\\_hashtable\\_size.h](#).

## 6.235 profiler\_list\_to\_slist.h File Reference

Diagnostics for list to slist.

### Namespaces

- namespace [\\_\\_gnu\\_profile](#)

## Functions

- void `__gnu_profile::__trace_list_to_slist_construct` (const void \*)
- void `__gnu_profile::__trace_list_to_slist_destruct` (const void \*)
- void `__gnu_profile::__trace_list_to_slist_init` ()
- void `__gnu_profile::__trace_list_to_slist_operation` (const void \*)
- void `__gnu_profile::__trace_list_to_slist_report` (FILE \*\_\_f, \_\_warning\_vector\_t &\_\_warnings)
- void `__gnu_profile::__trace_list_to_slist_rewind` (const void \*)

### 6.235.1 Detailed Description

Diagnostics for list to slist.

Definition in file [profiler\\_list\\_to\\_slist.h](#).

## 6.236 profiler\_list\_to\_vector.h File Reference

diagnostics for list to vector.

## Classes

- class [\\_\\_gnu\\_profile::\\_\\_list2vector\\_info](#)  
*A list-to-vector instrumentation line in the object table.*

## Namespaces

- namespace [\\_\\_gnu\\_profile](#)

## Functions

- void `__gnu_profile::__trace_list_to_vector_construct` (const void \*)
- void `__gnu_profile::__trace_list_to_vector_destruct` (const void \*)
- void `__gnu_profile::__trace_list_to_vector_init` ()
- void `__gnu_profile::__trace_list_to_vector_insert` (const void \*, std::size\_t, std::size\_t)
- void `__gnu_profile::__trace_list_to_vector_invalid_operator` (const void \*)
- void `__gnu_profile::__trace_list_to_vector_iterate` (const void \*, std::size\_t)
- void `__gnu_profile::__trace_list_to_vector_report` (FILE \*\_\_f, \_\_warning\_vector\_t &\_\_warnings)
- void `__gnu_profile::__trace_list_to_vector_resize` (const void \*, std::size\_t, std::size\_t)

### 6.236.1 Detailed Description

diagnostics for list to vector.

Definition in file [profiler\\_list\\_to\\_vector.h](#).

## 6.237 profiler\_map\_to\_unordered\_map.h File Reference

Diagnostics for map to unordered\_map.

### Classes

- class [\\_\\_gnu\\_profile::\\_\\_map2umap\\_info](#)  
*A map-to-unordered\_map instrumentation line in the object table.*
- class [\\_\\_gnu\\_profile::\\_\\_map2umap\\_stack\\_info](#)  
*A map-to-unordered\_map instrumentation line in the stack table.*
- class [\\_\\_gnu\\_profile::\\_\\_trace\\_map2umap](#)  
*Map-to-unordered\_map instrumentation producer.*

### Namespaces

- namespace [\\_\\_gnu\\_profile](#)

### Functions

- int [\\_\\_gnu\\_profile::\\_\\_log2](#) (std::size\_t \_\_size)
- float [\\_\\_gnu\\_profile::\\_\\_map\\_erase\\_cost](#) (std::size\_t \_\_size)
- float [\\_\\_gnu\\_profile::\\_\\_map\\_find\\_cost](#) (std::size\_t \_\_size)
- float [\\_\\_gnu\\_profile::\\_\\_map\\_insert\\_cost](#) (std::size\_t \_\_size)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_construct](#) (const void \*)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_destruct](#) (const void \*)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_erase](#) (const void \*, std::size\_t, std::size\_t)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_find](#) (const void \*, std::size\_t)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_init](#) ()

- void `__gnu_profile::__trace_map_to_unordered_map_insert` (const void \*, std::size\_t, std::size\_t)
- void `__gnu_profile::__trace_map_to_unordered_map_invalidate` (const void \*)
- void `__gnu_profile::__trace_map_to_unordered_map_iterate` (const void \*, std::size\_t)
- void `__gnu_profile::__trace_map_to_unordered_map_report` (FILE \*\_\_f, \_\_warning\_vector\_t &\_\_warnings)

### 6.237.1 Detailed Description

Diagnostics for map to unordered\_map.

Definition in file [profiler\\_map\\_to\\_unordered\\_map.h](#).

## 6.238 profiler\_node.h File Reference

Data structures to represent a single profiling event.

### Classes

- class [\\_\\_gnu\\_profile::\\_\\_object\\_info\\_base](#)  
*Base class for a line in the object table.*
- class [\\_\\_gnu\\_profile::\\_\\_stack\\_hash](#)  
*Hash function for summary trace using call stack as index.*
- class [\\_\\_gnu\\_profile::\\_\\_stack\\_info\\_base< \\_\\_object\\_info >](#)  
*Base class for a line in the stack table.*

### Namespaces

- namespace [\\_\\_gnu\\_profile](#)

### Typedefs

- typedef void \* [\\_\\_gnu\\_profile::\\_\\_instruction\\_address\\_t](#)
- typedef const void \* [\\_\\_gnu\\_profile::\\_\\_object\\_t](#)
- typedef std::vector< [\\_\\_instruction\\_address\\_t](#) > [\\_\\_gnu\\_profile::\\_\\_stack\\_npt](#)
- typedef [\\_\\_stack\\_npt](#) \* [\\_\\_gnu\\_profile::\\_\\_stack\\_t](#)



## Functions

- `__stack_t __gnu_profile::__get_stack ()`
- `std::size_t __gnu_profile::__size (__stack_t __stack)`
- `std::size_t __gnu_profile::__stack_max_depth ()`
- `void __gnu_profile::__write (FILE * __f, __stack_t __stack)`

### 6.238.1 Detailed Description

Data structures to represent a single profiling event.

Definition in file [profiler\\_node.h](#).

## 6.239 profiler\_state.h File Reference

Global profiler state.

## Namespaces

- namespace [\\_\\_gnu\\_profile](#)

## Enumerations

- enum `__state_type` { `__ON`, `__OFF`, `__INVALID` }

## Functions

- `bool __gnu_profile::__is_invalid ()`
- `bool __gnu_profile::__is_off ()`
- `bool __gnu_profile::__is_on ()`
- `bool __gnu_profile::__turn (__state_type __s)`
- `bool __gnu_profile::__turn_off ()`
- `bool __gnu_profile::__turn_on ()`
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA (__state_type, __-state, __INVALID)`

### 6.239.1 Detailed Description

Global profiler state.

Definition in file [profiler\\_state.h](#).

## 6.240 profiler\_trace.h File Reference

Data structures to represent profiling traces.

### Classes

- class [\\_\\_gnu\\_profile::\\_\\_trace\\_base< \\_\\_object\\_info, \\_\\_stack\\_info >](#)  
*Base class for all trace producers.*
- struct [\\_\\_gnu\\_profile::\\_\\_warning\\_data](#)  
*Representation of a warning.*

### Namespaces

- namespace [\\_\\_gnu\\_profile](#)

### Defines

- `#define _GLIBCXX_IMPL_UNORDERED_MAP`

### Typedefs

- `typedef std::vector< __cost_factor * > __gnu_profile::__cost_factor_vector`
- `typedef std::unordered_map< std::string, std::string > __gnu_profile::__env_t`
- `typedef std::vector< __warning_data > __gnu_profile::__warning_vector_t`

### Functions

- `std::size_t __gnu_profile::__env_to_size_t (const char *__env_var, std::size_t __default_value)`
- `__cost_factor_vector * & __gnu_profile::__get__cost_factors ()`
- `__env_t & __gnu_profile::__get__env ()`
- `__gnu_cxx::__mutex & __gnu_profile::__get__global_lock ()`
- `__cost_factor & __gnu_profile::__get__list_iterate_cost_factor ()`
- `__cost_factor & __gnu_profile::__get__list_resize_cost_factor ()`
- `__cost_factor & __gnu_profile::__get__list_shift_cost_factor ()`
- `__cost_factor & __gnu_profile::__get__map_erase_cost_factor ()`
- `__cost_factor & __gnu_profile::__get__map_find_cost_factor ()`
- `__cost_factor & __gnu_profile::__get__map_insert_cost_factor ()`

- `__cost_factor & __gnu_profile::__get__map_iterate_cost_factor ()`
- `__cost_factor & __gnu_profile::__get__umap_erase_cost_factor ()`
- `__cost_factor & __gnu_profile::__get__umap_find_cost_factor ()`
- `__cost_factor & __gnu_profile::__get__umap_insert_cost_factor ()`
- `__cost_factor & __gnu_profile::__get__umap_iterate_cost_factor ()`
- `__cost_factor & __gnu_profile::__get__vector_iterate_cost_factor ()`
- `__cost_factor & __gnu_profile::__get__vector_resize_cost_factor ()`
- `__cost_factor & __gnu_profile::__get__vector_shift_cost_factor ()`
- `__trace_hash_func *& __gnu_profile::__get__S_hash_func ()`
- `__trace_hashtable_size *& __gnu_profile::__get__S_hashtable_size ()`
- `__trace_list_to_slist *& __gnu_profile::__get__S_list_to_slist ()`
- `__trace_list_to_vector *& __gnu_profile::__get__S_list_to_vector ()`
- `__trace_map2umap *& __gnu_profile::__get__S_map2umap ()`
- `std::size_t & __gnu_profile::__get__S_max_mem ()`
- `std::size_t & __gnu_profile::__get__S_max_stack_depth ()`
- `std::size_t & __gnu_profile::__get__S_max_warn_count ()`
- `const char *& __gnu_profile::__get__S_trace_file_name ()`
- `__trace_vector_size *& __gnu_profile::__get__S_vector_size ()`
- `__trace_vector_to_list *& __gnu_profile::__get__S_vector_to_list ()`
- `int __gnu_profile::__log_magnitude (float __f)`
- `std::size_t __gnu_profile::__max_mem ()`
- `FILE * __gnu_profile::__open_output_file (const char *__extension)`
- `bool __gnu_profile::__profcxx_init ()`
- `void __gnu_profile::__profcxx_init_unconditional ()`
- `void __gnu_profile::__read_cost_factors ()`
- `void __gnu_profile::__report (void)`
- `void __gnu_profile::__set_cost_factors ()`
- `void __gnu_profile::__set_max_mem ()`
- `void __gnu_profile::__set_max_stack_trace_depth ()`
- `void __gnu_profile::__set_max_warn_count ()`
- `void __gnu_profile::__set_trace_path ()`
- `std::size_t __gnu_profile::__stack_max_depth ()`
- `void __gnu_profile::__trace_hash_func_init ()`
- `void __gnu_profile::__trace_hash_func_report (FILE *__f, __warning_vector_t & __warnings)`
- `void __gnu_profile::__trace_hashtable_size_init ()`
- `void __gnu_profile::__trace_hashtable_size_report (FILE *__f, __warning_vector_t & __warnings)`
- `void __gnu_profile::__trace_list_to_slist_init ()`
- `void __gnu_profile::__trace_list_to_slist_report (FILE *__f, __warning_vector_t & __warnings)`
- `void __gnu_profile::__trace_list_to_vector_init ()`

- void `__gnu_profile::__trace_list_to_vector_report` (FILE \*\_\_f, \_\_warning\_vector\_t &\_\_warnings)
- void `__gnu_profile::__trace_map_to_unordered_map_init` ()
- void `__gnu_profile::__trace_map_to_unordered_map_report` (FILE \*\_\_f, \_\_warning\_vector\_t &\_\_warnings)
- void `__gnu_profile::__trace_vector_size_init` ()
- void `__gnu_profile::__trace_vector_size_report` (FILE \*, \_\_warning\_vector\_t &)
- void `__gnu_profile::__trace_vector_to_list_init` ()
- void `__gnu_profile::__trace_vector_to_list_report` (FILE \*, \_\_warning\_vector\_t &)
- void `__gnu_profile::__write_cost_factors` ()

#### 6.240.1 Detailed Description

Data structures to represent profiling traces.

Definition in file [profiler\\_trace.h](#).

## 6.241 profiler\_vector\_size.h File Reference

Collection of vector size traces.

### Classes

- class [\\_\\_gnu\\_profile::\\_\\_trace\\_vector\\_size](#)  
*Hashtable size instrumentation trace producer.*

### Namespaces

- namespace [\\_\\_gnu\\_profile](#)

### Functions

- void `__gnu_profile::__trace_vector_size_construct` (const void \*, std::size\_t)
- void `__gnu_profile::__trace_vector_size_destruct` (const void \*, std::size\_t, std::size\_t)
- void `__gnu_profile::__trace_vector_size_init` ()
- void `__gnu_profile::__trace_vector_size_report` (FILE \*, \_\_warning\_vector\_t &)
- void `__gnu_profile::__trace_vector_size_resize` (const void \*, std::size\_t, std::size\_t)

### 6.241.1 Detailed Description

Collection of vector size traces.

Definition in file [profiler\\_vector\\_size.h](#).

## 6.242 profiler\_vector\_to\_list.h File Reference

diagnostics for vector to list.

### Classes

- class [\\_\\_gnu\\_profile::\\_\\_trace\\_vector\\_to\\_list](#)  
*Vector-to-list instrumentation producer.*
- class [\\_\\_gnu\\_profile::\\_\\_vector2list\\_info](#)  
*A vector-to-list instrumentation line in the object table.*
- class [\\_\\_gnu\\_profile::\\_\\_vector2list\\_stack\\_info](#)  
*A vector-to-list instrumentation line in the stack table.*

### Namespaces

- namespace [\\_\\_gnu\\_profile](#)

### Functions

- void [\\_\\_gnu\\_profile::\\_\\_trace\\_vector\\_to\\_list\\_construct](#) (const void \*)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_vector\\_to\\_list\\_destruct](#) (const void \*)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_vector\\_to\\_list\\_find](#) (const void \*, std::size\_t)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_vector\\_to\\_list\\_init](#) ()
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_vector\\_to\\_list\\_insert](#) (const void \*, std::size\_t, std::size\_t)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_vector\\_to\\_list\\_invalid\\_operator](#) (const void \*)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_vector\\_to\\_list\\_iterate](#) (const void \*, std::size\_t)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_vector\\_to\\_list\\_report](#) (FILE \*, \_\_warning\_vector\_t &)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_vector\\_to\\_list\\_resize](#) (const void \*, std::size\_t, std::size\_t)

### 6.242.1 Detailed Description

diagnostics for vector to list.

Definition in file [profiler\\_vector\\_to\\_list.h](#).

## 6.243 queue File Reference

### 6.243.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [queue](#).

## 6.244 queue.h File Reference

Lock-free double-ended queue. This file is a GNU parallel extension to the Standard C++ Library.

### Classes

- class [\\_\\_gnu\\_parallel::\\_RestrictedBoundedConcurrentQueue<\\_Tp>](#)  
*Double-ended queue of bounded size, allowing lock-free atomic access. [push\\_front\(\)](#) and [pop\\_front\(\)](#) must not be called concurrently to each other; while [pop\\_back\(\)](#) can be called concurrently at all times. [empty\(\)](#), [size\(\)](#), and [top\(\)](#) are intentionally not provided. Calling them would not make sense in a concurrent setting.*

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Defines

- [#define \\_GLIBCXX\\_VOLATILE](#)

### 6.244.1 Detailed Description

Lock-free double-ended queue. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [queue.h](#).

## 6.244.2 Define Documentation

### 6.244.2.1 #define \_GLIBCXX\_VOLATILE

Decide whether to declare certain variable volatile in this file.

Definition at line 40 of file queue.h.

## 6.245 quicksort.h File Reference

Implementation of a unbalanced parallel quicksort (in-place). This file is a GNU parallel extension to the Standard C++ Library.

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _RAIter, typename _Compare >  
void \_\_gnu\_parallel::\_\_parallel\_sort\_qs (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >  
void \_\_gnu\_parallel::\_\_parallel\_sort\_qs\_conquer (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >  
std::iterator_traits< _RAIter >::difference_type \_\_gnu\_parallel::\_\_parallel\_sort\_qs\_divide (_RAIter __begin, _RAIter __end, _Compare __comp, typename std::iterator_traits< _RAIter >::difference_type __pivot_rank, typename std::iterator_traits< _RAIter >::difference_type __num_samples, _ThreadIndex __num_threads)`

### 6.245.1 Detailed Description

Implementation of a unbalanced parallel quicksort (in-place). This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [quicksort.h](#).

## 6.246 random File Reference

### 6.246.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [random](#).

## 6.247 random.h File Reference

### Classes

- class [std::bernoulli\\_distribution](#)  
*A Bernoulli random number distribution.*
- struct [std::bernoulli\\_distribution::param\\_type](#)
- class [std::binomial\\_distribution<\\_IntType>](#)  
*A discrete binomial random number distribution.*
- struct [std::binomial\\_distribution<\\_IntType>::param\\_type](#)
- class [std::cauchy\\_distribution<\\_RealType>](#)  
*A [cauchy\\_distribution](#) random number distribution.*
- struct [std::cauchy\\_distribution<\\_RealType>::param\\_type](#)
- class [std::chi\\_squared\\_distribution<\\_RealType>](#)  
*A [chi\\_squared\\_distribution](#) random number distribution.*
- struct [std::chi\\_squared\\_distribution<\\_RealType>::param\\_type](#)
- class [std::discard\\_block\\_engine<\\_RandomNumberEngine, \\_\\_p, \\_\\_r>](#)
- class [std::discrete\\_distribution<\\_IntType>](#)  
*A [discrete\\_distribution](#) random number distribution.*
- struct [std::discrete\\_distribution<\\_IntType>::param\\_type](#)
- class [std::exponential\\_distribution<\\_RealType>](#)  
*An exponential continuous distribution for random numbers.*
- struct [std::exponential\\_distribution<\\_RealType>::param\\_type](#)
- class [std::extreme\\_value\\_distribution<\\_RealType>](#)  
*A [extreme\\_value\\_distribution](#) random number distribution.*
- struct [std::extreme\\_value\\_distribution<\\_RealType>::param\\_type](#)
- class [std::fisher\\_f\\_distribution<\\_RealType>](#)  
*A [fisher\\_f\\_distribution](#) random number distribution.*



- struct `std::fisher_f_distribution<_RealType>::param_type`
- class `std::gamma_distribution<_RealType>`  
*A gamma continuous distribution for random numbers.*
- struct `std::gamma_distribution<_RealType>::param_type`
- class `std::geometric_distribution<_IntType>`  
*A discrete geometric random number distribution.*
- struct `std::geometric_distribution<_IntType>::param_type`
- class `std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>`  
*A model of a linear congruential random number generator.*
- class `std::linear_congruential_engine<_UIntType, __a, __c, __m>`  
*A model of a linear congruential random number generator.*
- class `std::lognormal_distribution<_RealType>`  
*A lognormal\_distribution random number distribution.*
- struct `std::lognormal_distribution<_RealType>::param_type`
- class `std::negative_binomial_distribution<_IntType>`  
*A negative\_binomial\_distribution random number distribution.*
- struct `std::negative_binomial_distribution<_IntType>::param_type`
- class `std::normal_distribution<_RealType>`  
*A normal continuous distribution for random numbers.*
- struct `std::normal_distribution<_RealType>::param_type`
- class `std::piecewise_constant_distribution<_RealType>`  
*A piecewise\_constant\_distribution random number distribution.*
- struct `std::piecewise_constant_distribution<_RealType>::param_type`
- class `std::piecewise_linear_distribution<_RealType>`  
*A piecewise\_linear\_distribution random number distribution.*
- struct `std::piecewise_linear_distribution<_RealType>::param_type`
- class `std::poisson_distribution<_IntType>`  
*A discrete Poisson random number distribution.*
- struct `std::poisson_distribution<_IntType>::param_type`
- class `std::random_device`
- class `std::seed_seq`  
*The seed\_seq class generates sequences of seeds for random number generators.*

- class [std::shuffle\\_order\\_engine< \\_RandomNumberEngine, \\_\\_k >](#)  
*Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits \_\_w.*
- class [std::student\\_t\\_distribution< \\_RealType >](#)  
*A [student\\_t\\_distribution](#) random number distribution.*
- struct [std::student\\_t\\_distribution< \\_RealType >::param\\_type](#)
- class [std::uniform\\_int\\_distribution< \\_IntType >](#)  
*Uniform discrete distribution for random numbers. A discrete random distribution on the range[*min*, *max*] with equal probability throughout the range.*
- struct [std::uniform\\_int\\_distribution< \\_IntType >::param\\_type](#)
- class [std::uniform\\_real\\_distribution< \\_RealType >](#)  
*Uniform continuous distribution for random numbers.*
- struct [std::uniform\\_real\\_distribution< \\_RealType >::param\\_type](#)
- class [std::weibull\\_distribution< \\_RealType >](#)  
*A [weibull\\_distribution](#) random number distribution.*
- struct [std::weibull\\_distribution< \\_RealType >::param\\_type](#)

## Namespaces

- namespace [std](#)
- namespace [std::\\_\\_detail](#)

## Typedefs

- typedef minstd\_rand0 [std::default\\_random\\_engine](#)
- typedef shuffle\_order\_engine< minstd\_rand0, 256 > [std::knuth\\_b](#)
- typedef linear\_congruential\_engine< uint\_fast32\_t, 48271UL, 0UL, 2147483647UL > [std::minstd\\_rand](#)
- typedef linear\_congruential\_engine< uint\_fast32\_t, 16807UL, 0UL, 2147483647UL > [std::minstd\\_rand0](#)
- typedef mersenne\_twister\_engine< uint\_fast32\_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL > [std::mt19937](#)
- typedef mersenne\_twister\_engine< uint\_fast64\_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000ULL, 37, 0xff7eee0000000000ULL, 43, 6364136223846793005ULL > [std::mt19937\\_64](#)
- typedef discard\_block\_engine< ranlux24\_base, 223, 23 > [std::ranlux24](#)

- typedef subtract\_with\_carry\_engine< uint\_fast32\_t, 24, 10, 24 > **std::ranlux24\_base**
- typedef discard\_block\_engine< ranlux48\_base, 389, 11 > **std::ranlux48**
- typedef subtract\_with\_carry\_engine< uint\_fast64\_t, 48, 5, 12 > **std::ranlux48\_base**

## Functions

- template<typename \_RealType, size\_t \_\_bits, typename \_UniformRandomNumberGenerator >  
\_RealType [std::generate\\_canonical](#) (\_UniformRandomNumberGenerator &\_\_g)
- template<typename \_UIntType, \_UIntType \_\_a, \_UIntType \_\_c, \_UIntType \_\_m>  
bool [std::operator!=](#) (const [std::linear\\_congruential\\_engine](#)< \_UIntType, \_\_a, \_\_c, \_\_m > &\_\_lhs, const [std::linear\\_congruential\\_engine](#)< \_UIntType, \_\_a, \_\_c, \_\_m > &\_\_rhs)
- template<typename \_UIntType, size\_t \_\_w, size\_t \_\_s, size\_t \_\_r>  
bool [std::operator!=](#) (const [std::subtract\\_with\\_carry\\_engine](#)< \_UIntType, \_\_w, \_\_s, \_\_r > &\_\_lhs, const [std::subtract\\_with\\_carry\\_engine](#)< \_UIntType, \_\_w, \_\_s, \_\_r > &\_\_rhs)
- template<typename \_RandomNumberEngine, size\_t \_\_k>  
bool [std::operator!=](#) (const [std::shuffle\\_order\\_engine](#)< \_RandomNumberEngine, \_\_k > &\_\_lhs, const [std::shuffle\\_order\\_engine](#)< \_RandomNumberEngine, \_\_k > &\_\_rhs)
- template<typename \_IntType >  
bool [std::operator!=](#) (const [std::geometric\\_distribution](#)< \_IntType > &\_\_d1, const [std::geometric\\_distribution](#)< \_IntType > &\_\_d2)
- template<typename \_RealType >  
bool [std::operator!=](#) (const [std::normal\\_distribution](#)< \_RealType > &\_\_d1, const [std::normal\\_distribution](#)< \_RealType > &\_\_d2)
- template<typename \_RealType >  
bool [std::operator!=](#) (const [std::lognormal\\_distribution](#)< \_RealType > &\_\_d1, const [std::lognormal\\_distribution](#)< \_RealType > &\_\_d2)
- template<typename \_IntType >  
bool [std::operator!=](#) (const [std::negative\\_binomial\\_distribution](#)< \_IntType > &\_\_d1, const [std::negative\\_binomial\\_distribution](#)< \_IntType > &\_\_d2)
- template<typename \_IntType >  
bool [std::operator!=](#) (const [std::poisson\\_distribution](#)< \_IntType > &\_\_d1, const [std::poisson\\_distribution](#)< \_IntType > &\_\_d2)
- template<typename \_RealType >  
bool [std::operator!=](#) (const [std::gamma\\_distribution](#)< \_RealType > &\_\_d1, const [std::gamma\\_distribution](#)< \_RealType > &\_\_d2)
- template<typename \_RealType >  
bool [std::operator!=](#) (const [std::exponential\\_distribution](#)< \_RealType > &\_\_d1, const [std::exponential\\_distribution](#)< \_RealType > &\_\_d2)

- `template<typename _RandomNumberEngine, size_t __p, size_t __r>`  
`bool std::operator!= (const std::discard_block_engine< _-`  
`RandomNumberEngine, __p, __r > &__lhs, const std::discard_block_engine<`  
`_RandomNumberEngine, __p, __r > &__rhs)`
- `template<typename _IntType >`  
`bool std::operator!= (const std::uniform_int_distribution< _IntType > &__d1,`  
`const std::uniform_int_distribution< _IntType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::chi_squared_distribution< _RealType > &__d1,`  
`const std::chi_squared_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::weibull_distribution< _RealType > &__d1,`  
`const std::weibull_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::cauchy_distribution< _RealType > &__d1,`  
`const std::cauchy_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::extreme_value_distribution< _RealType > &__-`  
`__d1, const std::extreme_value_distribution< _RealType > &__d2)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a,`  
`size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _-`  
`_UIntType __f>`  
`bool std::operator!= (const std::mersenne_twister_engine< _UIntType, __w, _-`  
`__n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__lhs, const`  
`std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d,`  
`__s, __b, __t, __c, __l, __f > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType >`  
`bool std::operator!= (const std::independent_bits_engine< _-`  
`RandomNumberEngine, __w, _UIntType > &__lhs, const std::independent_-`  
`bits_engine< _RandomNumberEngine, __w, _UIntType > &__rhs)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::fisher_f_distribution< _RealType > &__d1,`  
`const std::fisher_f_distribution< _RealType > &__d2)`
- `template<typename _IntType >`  
`bool std::operator!= (const std::discrete_distribution< _IntType > &__d1, const`  
`std::discrete_distribution< _IntType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::student_t_distribution< _RealType > &__d1,`  
`const std::student_t_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::piecewise_constant_distribution< _RealType >`  
`&__d1, const std::piecewise_constant_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::piecewise_linear_distribution< _RealType >`  
`&__d1, const std::piecewise_linear_distribution< _RealType > &__d2)`

- `template<typename _IntType >`  
`bool std::operator!= (const std::uniform_real_distribution< _IntType > &__d1,`  
`const std::uniform_real_distribution< _IntType > &__d2)`
- `bool std::operator!= (const std::bernoulli_distribution &__d1, const`  
`std::bernoulli_distribution &__d2)`
- `template<typename _IntType >`  
`bool std::operator!= (const std::binomial_distribution< _IntType > &__d1,`  
`const std::binomial_distribution< _IntType > &__d2)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_-`  
`ostream< _CharT, _Traits > &, const std::cauchy_distribution< _RealType >`  
`&)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_-`  
`ostream< _CharT, _Traits > &, const std::geometric_distribution< _IntType >`  
`&)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_-`  
`ostream< _CharT, _Traits > &, const std::weibull_distribution< _RealType >`  
`&)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_-`  
`ostream< _CharT, _Traits > &, const std::exponential_distribution< _RealType`  
`> &)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_-`  
`ostream< _CharT, _Traits > &, const std::uniform_int_distribution< _IntType`  
`> &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_-`  
`ostream< _CharT, _Traits > &, const std::extreme_value_distribution< _`  
`RealType > &)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT`  
`, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_-`  
`ostream< _CharT, _Traits > &__os, const std::independent_bits_engine< _`  
`RandomNumberEngine, __w, _UIntType > &__x)`
- `template<typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_-`  
`ostream< _CharT, _Traits > &, const std::bernoulli_distribution &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_-`  
`ostream< _CharT, _Traits > &, const std::uniform_real_distribution< _`  
`RealType > &)`

- `bool std::operator== (const std::bernoulli_distribution &__d1, const std::bernoulli_distribution &__d2)`
- `template<typename _RealType >  
bool std::operator== (const std::exponential_distribution< _RealType > &__d1, const std::exponential_distribution< _RealType > &__d2)`
- `template<typename _IntType >  
bool std::operator== (const std::uniform_int_distribution< _IntType > &__d1, const std::uniform_int_distribution< _IntType > &__d2)`
- `template<typename _IntType >  
bool std::operator== (const std::geometric_distribution< _IntType > &__d1, const std::geometric_distribution< _IntType > &__d2)`
- `template<typename _IntType >  
bool std::operator== (const std::uniform_real_distribution< _IntType > &__d1, const std::uniform_real_distribution< _IntType > &__d2)`
- `template<typename _IntType >  
bool std::operator== (const std::discrete_distribution< _IntType > &__d1, const std::discrete_distribution< _IntType > &__d2)`
- `template<typename _RealType >  
bool std::operator== (const std::extreme_value_distribution< _RealType > &__d1, const std::extreme_value_distribution< _RealType > &__d2)`
- `template<typename _RealType >  
bool std::operator== (const std::weibull_distribution< _RealType > &__d1, const std::weibull_distribution< _RealType > &__d2)`
- `template<typename _RealType >  
bool std::operator== (const std::cauchy_distribution< _RealType > &__d1, const std::cauchy_distribution< _RealType > &__d2)`
- `template<typename _RealType >  
bool std::operator== (const std::piecewise_linear_distribution< _RealType > &__d1, const std::piecewise_linear_distribution< _RealType > &__d2)`
- `template<typename _RealType >  
bool std::operator== (const std::piecewise_constant_distribution< _RealType > &__d1, const std::piecewise_constant_distribution< _RealType > &__d2)`
- `template<typename _IntType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_int_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::exponential_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::weibull_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::extreme_value_distribution< _RealType > &)`

- `template<typename _IntType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<  
_CharT, _Traits > &, std::geometric_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<  
_CharT, _Traits > &, std::uniform_real_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<  
_CharT, _Traits > &, std::cauchy_distribution< _RealType > &)`
- `template<typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<  
_CharT, _Traits > & __is, std::bernoulli_distribution & __x)`

### 6.247.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<random>`.

Definition in file [random.h](#).

## 6.248 random.tcc File Reference

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_detail](#)

### Functions

- `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation >  
_OutputIterator std::__detail::__transform (_InputIterator __first, _-  
InputIterator __last, _OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator >  
_RealType std::generate_canonical (_UniformRandomNumberGenerator & __g)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT  
, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_-  
ostream< _CharT, _Traits > & __os, const linear_congruential_engine< _-  
UIntType, __a, __c, __m > & __lcr)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a,  
size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _-  
UIntType __f, typename _CharT, typename _Traits >`

```
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__x)
```

- template<typename \_RandomNumberEngine, size\_t \_\_p, size\_t \_\_r, typename \_CharT, typename \_Traits >  
std::basic\_ostream< \_CharT, \_Traits > & std::operator<< (std::basic\_ostream< \_CharT, \_Traits > &\_\_os, const discard\_block\_engine< \_RandomNumberEngine, \_\_p, \_\_r > &\_\_x)
- template<typename \_CharT, typename \_Traits >  
std::basic\_ostream< \_CharT, \_Traits > & std::operator<< (std::basic\_ostream< \_CharT, \_Traits > &, const std::bernoulli\_distribution &)
- template<typename \_RealType, typename \_CharT, typename \_Traits >  
std::basic\_ostream< \_CharT, \_Traits > & std::operator<< (std::basic\_ostream< \_CharT, \_Traits > &\_\_os, const chi\_squared\_distribution< \_RealType > &\_\_x)
- template<typename \_IntType, typename \_CharT, typename \_Traits >  
std::basic\_ostream< \_CharT, \_Traits > & std::operator<< (std::basic\_ostream< \_CharT, \_Traits > &, const std::geometric\_distribution< \_IntType > &)
- template<typename \_RealType, typename \_CharT, typename \_Traits >  
std::basic\_ostream< \_CharT, \_Traits > & std::operator<< (std::basic\_ostream< \_CharT, \_Traits > &, const std::cauchy\_distribution< \_RealType > &)
- template<typename \_IntType, typename \_CharT, typename \_Traits >  
std::basic\_ostream< \_CharT, \_Traits > & std::operator<< (std::basic\_ostream< \_CharT, \_Traits > &\_\_os, const negative\_binomial\_distribution< \_IntType > &\_\_x)
- template<typename \_RealType, typename \_CharT, typename \_Traits >  
std::basic\_ostream< \_CharT, \_Traits > & std::operator<< (std::basic\_ostream< \_CharT, \_Traits > &\_\_os, const fisher\_f\_distribution< \_RealType > &\_\_x)
- template<typename \_RealType, typename \_CharT, typename \_Traits >  
std::basic\_ostream< \_CharT, \_Traits > & std::operator<< (std::basic\_ostream< \_CharT, \_Traits > &\_\_os, const student\_t\_distribution< \_RealType > &\_\_x)
- template<typename \_RandomNumberEngine, size\_t \_\_k, typename \_CharT, typename \_Traits >  
std::basic\_ostream< \_CharT, \_Traits > & std::operator<< (std::basic\_ostream< \_CharT, \_Traits > &\_\_os, const shuffle\_order\_engine< \_RandomNumberEngine, \_\_k > &\_\_x)
- template<typename \_IntType, typename \_CharT, typename \_Traits >  
std::basic\_ostream< \_CharT, \_Traits > & std::operator<< (std::basic\_ostream< \_CharT, \_Traits > &\_\_os, const poisson\_distribution< \_IntType > &\_\_x)



- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const gamma_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::weibull_distribution< _RealType > &)`
- `template<typename _IntType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const binomial_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::extreme_value_distribution< _RealType > &)`
- `template<typename _IntType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const discrete_distribution< _IntType > &__x)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const subtract_with_carry_engine< _UIntType, __w, __s, __r > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::exponential_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const piecewise_constant_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const piecewise_linear_distribution< _RealType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_int_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_real_distribution< _RealType > &)`

- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const normal_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const lognormal_distribution< _RealType > &__x)`
- `template<typename _RealType >`  
`bool std::operator== (const std::normal_distribution< _RealType > &__d1, const std::normal_distribution< _RealType > &__d2)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, piecewise_constant_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::exponential_distribution< _RealType > &)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, discard_block_engine< _RandomNumberEngine, __p, __r > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, student_t_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, gamma_distribution< _RealType > &__x)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::extreme_value_distribution< _RealType > &)`
- `template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, shuffle_order_engine< _RandomNumberEngine, __k > &__x)`

- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, binomial_distribution< _IntType > &__x)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, linear_congruential_engine< _UIntType, __a, __c, __m > &__lcr)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, normal_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_real_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, lognormal_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, chi_squared_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, fisher_f_distribution< _RealType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, negative_binomial_distribution< _IntType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, poisson_distribution< _IntType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, discrete_distribution< _IntType > &__x)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, subtract_with_carry_engine< _UIntType, __w, __s, __r > &__x)`

- `template<typename _IntType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<  
_CharT, _Traits > &, std::uniform_int_distribution< _IntType > &)`
- `template<typename _IntType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<  
_CharT, _Traits > &, std::geometric_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<  
_CharT, _Traits > &, std::weibull_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<  
_CharT, _Traits > & __is, piecewise_linear_distribution< _RealType  
> & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<  
_CharT, _Traits > &, std::cauchy_distribution< _RealType > &)`

### 6.248.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<random>`.

Definition in file [random.tcc](#).

## 6.249 random\_number.h File Reference

Random number generator based on the Mersenne twister. This file is a GNU parallel extension to the Standard C++ Library.

### Classes

- class [\\_\\_gnu\\_parallel::\\_RandomNumber](#)  
*Random number generator, based on the Mersenne twister.*

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### 6.249.1 Detailed Description

Random number generator based on the Mersenne twister. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [random\\_number.h](#).

## 6.250 random\_shuffle.h File Reference

Parallel implementation of `std::random_shuffle()`. This file is a GNU parallel extension to the Standard C++ Library.

### Classes

- struct [\\_\\_gnu\\_parallel::\\_DRandomShufflingGlobalData<\\_RAIter>](#)  
*Data known to every thread participating in [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\(\)](#).*
- struct [\\_\\_gnu\\_parallel::\\_DRSSorterPU<\\_RAIter, \\_RandomNumberGenerator>](#)  
*Local data for a thread participating in [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\(\)](#).*

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Typedefs

- typedef unsigned short [\\_\\_gnu\\_parallel::\\_BinIndex](#)

### Functions

- `template<typename _RAIter, typename _RandomNumberGenerator>`  
`void \_\_gnu\_parallel::\_\_parallel\_random\_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator __rng=_RandomNumber())`
- `template<typename _RAIter, typename _RandomNumberGenerator>`  
`void \_\_gnu\_parallel::\_\_parallel\_random\_shuffle\_drs (_RAIter __begin, _RAIter __end, typename std::iterator_traits<_RAIter>::difference_type __n, _ThreadIndex __num_threads, _RandomNumberGenerator &__rng)`
- `template<typename _RAIter, typename _RandomNumberGenerator>`  
`void \_\_gnu\_parallel::\_\_parallel\_random\_shuffle\_drs\_pu (_DRSSorterPU<_RAIter, _RandomNumberGenerator> *__pus)`
- `template<typename _RandomNumberGenerator>`  
`int \_\_gnu\_parallel::\_\_random\_number\_pow2 (int __logp, _RandomNumberGenerator &__rng)`

- `template<typename _Tp >`  
`_Tp __gnu_parallel::__round_up_to_pow2 (_Tp __x)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void __gnu_parallel::__sequential_random_shuffle (_RAIter __begin, _RAIter`  
`__end, _RandomNumberGenerator &__rng)`

### 6.250.1 Detailed Description

Parallel implementation of `std::random_shuffle()`. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [random\\_shuffle.h](#).

## 6.251 range\_access.h File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<class _Container >`  
`auto std::begin (_Container &__cont)-> decltype(__cont.begin())`
- `template<class _Container >`  
`auto std::begin (const _Container &__cont)-> decltype(__cont.begin())`
- `template<class _Tp, size_t _Nm>`  
`_Tp * std::begin (_Tp(&__arr)[_Nm])`
- `template<class _Container >`  
`auto std::end (const _Container &__cont)-> decltype(__cont.end())`
- `template<class _Container >`  
`auto std::end (_Container &__cont)-> decltype(__cont.end())`
- `template<class _Tp, size_t _Nm>`  
`_Tp * std::end (_Tp(&__arr)[_Nm])`

### 6.251.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

Definition in file [range\\_access.h](#).

## 6.252 **ratio** File Reference

### Classes

- struct [std::ratio< \\_Num, \\_Den >](#)  
*Provides compile-time rational arithmetic.*
- struct [std::ratio\\_add< \\_R1, \\_R2 >](#)  
*ratio\_add*
- struct [std::ratio\\_divide< \\_R1, \\_R2 >](#)  
*ratio\_divide*
- struct [std::ratio\\_equal< \\_R1, \\_R2 >](#)  
*ratio\_equal*
- struct [std::ratio\\_multiply< \\_R1, \\_R2 >](#)  
*ratio\_multiply*
- struct [std::ratio\\_not\\_equal< \\_R1, \\_R2 >](#)  
*ratio\_not\_equal*
- struct [std::ratio\\_subtract< \\_R1, \\_R2 >](#)  
*ratio\_subtract*

### Namespaces

- namespace [std](#)

#### 6.252.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [ratio](#).

## 6.253 **rb\_tree** File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::rb\\_tree< \\_Key, \\_Value, \\_KeyOfValue, \\_Compare, \\_Alloc >](#)

## Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### 6.253.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [rb\\_tree](#).

## 6.254 rc\_string\_base.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::\\_\\_rc\\_string\\_base<\\_CharT, \\_Traits, \\_Alloc>](#)

## Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### 6.254.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include [<ext/vstring.h>](#).

Definition in file [rc\\_string\\_base.h](#).

## 6.255 regex File Reference

### 6.255.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [regex](#).

## 6.256 regex.h File Reference

### Classes

- class [std::basic\\_regex<\\_Ch\\_type, \\_Rx\\_traits>](#)
- class [std::match\\_results<\\_Bi\\_iter, \\_Allocator>](#)

*The results of a match or search operation.*



- class `std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >`
- class `std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >`
- struct `std::regex_traits< _Ch_type >`

*Describes aspects of a regular expression.*

- class `std::sub_match< _BiIter >`

## Namespaces

- namespace `std`

## Typedefs

- typedef `match_results< const char * >` **`std::cmatch`**
- typedef `regex_iterator< const char * >` **`std::cregex_iterator`**
- typedef `regex_token_iterator< const char * >` **`std::cregex_token_iterator`**
- typedef `sub_match< const char * >` **`std::csub_match`**
- typedef `basic_regex< char >` **`std::regex`**
- typedef `match_results< string::const_iterator >` **`std::smatch`**
- typedef `regex_iterator< string::const_iterator >` **`std::sregex_iterator`**
- typedef `regex_token_iterator< string::const_iterator >` **`std::sregex_token_iterator`**
- typedef `sub_match< string::const_iterator >` **`std::ssub_match`**
- typedef `match_results< const wchar_t * >` **`std::wcmatch`**
- typedef `regex_iterator< const wchar_t * >` **`std::wcregex_iterator`**
- typedef `regex_token_iterator< const wchar_t * >` **`std::wcregex_token_iterator`**
- typedef `sub_match< const wchar_t * >` **`std::wcs_sub_match`**
- typedef `basic_regex< wchar_t >` **`std::wregex`**
- typedef `match_results< wstring::const_iterator >` **`std::wsmatch`**
- typedef `regex_iterator< wstring::const_iterator >` **`std::wsregex_iterator`**
- typedef `regex_token_iterator< wstring::const_iterator >` **`std::wsregex_token_iterator`**
- typedef `sub_match< wstring::const_iterator >` **`std::wssub_match`**

## Functions

- template<typename `_Bi_iter` >  
const `sub_match< _Bi_iter >` & **`std::__unmatched_sub()`**
- template<typename `_BiIter` >  
bool **`std::operator!=`** (const `sub_match< _BiIter >` & `__lhs`, const `sub_match< _BiIter >` & `__rhs`)

- `template<typename _Bi_iter >`  
`bool std::operator!= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator!= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool std::operator!= (const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator!= (typename iterator_traits< _Bi_iter >::value_type const * __lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Allocator >`  
`bool std::operator!= (const match_results< _Bi_iter, _Allocator > &__m1, const match_results< _Bi_iter, _Allocator > &__m2)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool std::operator!= (const sub_match< _Bi_iter > &__lhs, const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator!= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const * __rhs)`
- `template<typename _Bilter >`  
`bool std::operator< (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`  
`bool std::operator< (const sub_match< _Bi_iter > &__lhs, const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator< (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool std::operator< (const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator< (typename iterator_traits< _Bi_iter >::value_type const * __lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const * __rhs)`

- `template<typename _Ch_type, typename _Ch_traits, typename _Bi_iter >`  
`basic_ostream< _Ch_type, _Ch_traits > & std::operator<< (basic_ostream<`  
`_Ch_type, _Ch_traits > &__os, const sub_match< _Bi_iter > &__m)`
- `template<typename _Bi_iter >`  
`bool std::operator<= (const sub_match< _Bi_iter > &__lhs, typename`  
`iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator<= (typename iterator_traits< _Bi_iter >::value_type const`  
`&__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`  
`bool std::operator<= (const sub_match< _Bi_iter > &__lhs, const basic_`  
`string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_`  
`alloc > &__rhs)`
- `template<typename _Bilter >`  
`bool std::operator<= (const sub_match< _Bilter > &__lhs, const sub_match<`  
`_Bilter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator<= (typename iterator_traits< _Bi_iter >::value_type const`  
`*__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool std::operator<= (const basic_string< typename iterator_traits< _Bi_iter`  
`>::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter >`  
`&__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator<= (const sub_match< _Bi_iter > &__lhs, typename`  
`iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool std::operator== (const sub_match< _Bi_iter > &__lhs, const basic_`  
`string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_`  
`alloc > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool std::operator== (const basic_string< typename iterator_traits< _Bi_iter`  
`>::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter >`  
`&__rhs)`
- `template<typename _Bi_iter, typename _Allocator >`  
`bool std::operator== (const match_results< _Bi_iter, _Allocator > &__m1,`  
`const match_results< _Bi_iter, _Allocator > &__m2)`
- `template<typename _Bi_iter >`  
`bool std::operator== (typename iterator_traits< _Bi_iter >::value_type const`  
`&__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator== (typename iterator_traits< _Bi_iter >::value_type const`  
`*__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_`  
`traits< _Bi_iter >::value_type const &__rhs)`

- `template<typename _Bilter >`  
`bool std::operator== (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool std::operator> (const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bilter >`  
`bool std::operator> (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator> (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator> (typename iterator_traits< _Bi_iter >::value_type const * __lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`  
`bool std::operator> (const sub_match< _Bi_iter > &__lhs, const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator>= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator>= (typename iterator_traits< _Bi_iter >::value_type const * __lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool std::operator>= (const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bilter >`  
`bool std::operator>= (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`  
`bool std::operator>= (const sub_match< _Bi_iter > &__lhs, const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__rhs)`

- `template<typename _Bi_iter >`  
`bool std::operator>= (const sub_match< _Bi_iter > &__lhs, typename`  
`iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter >`  
`bool std::operator>= (typename iterator_traits< _Bi_iter >::value_type const`  
`&__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Ch_type, typename _Rx_traits >`  
`void std::swap (basic_regex< _Ch_type, _Rx_traits > &__lhs, basic_regex< _`  
`Ch_type, _Rx_traits > &__rhs)`
- `template<typename _Bi_iter, typename _Allocator >`  
`void std::swap (match_results< _Bi_iter, _Allocator > &__lhs, match_results<`  
`_Bi_iter, _Allocator > &__rhs)`

### Matching, Searching, and Replacing

- `template<typename _Bi_iter, typename _Allocator, typename _Ch_type, typename _Rx_traits`  
`>`  
`bool std::regex_match (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter,`  
`_Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re,`  
`regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >`  
`bool std::regex_match (_Bi_iter __first, _Bi_iter __last, const basic_`  
`regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type _`  
`__flags=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Allocator, typename _Rx_traits >`  
`bool std::regex_match (const _Ch_type *__s, match_results< const _Ch_type`  
`*, _Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re,`  
`regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Allocator, typename _Ch_`  
`type, typename _Rx_traits >`  
`bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc`  
`> &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _`  
`Ch_alloc >::const_iterator, _Allocator > &__m, const basic_regex< _Ch_`  
`type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_`  
`constants::match_default)`
- `template<typename _Ch_type, class _Rx_traits >`  
`bool std::regex_match (const _Ch_type *__s, const basic_regex< _Ch_`  
`type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_`  
`constants::match_default)`
- `template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _`  
`Rx_traits >`  
`bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Str_`  
`allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__re,`  
`regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Allocator, typename _Ch_type, typename _Rx_traits`  
`>`

```

bool std::regex_search (_Bi_iter __first, _Bi_iter __last, match_results< _-
Bi_iter, _Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits >
&__re, regex_constants::match_flag_type __flags=regex_constants::match-
default)
• template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >
bool std::regex_search (_Bi_iter __first, _Bi_iter __last, const basic_-
regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type _-
__flags=regex_constants::match_default)
• template<typename _Ch_type, class _Allocator, class _Rx_traits >
bool std::regex_search (const _Ch_type *__s, match_results< const _Ch_-
type *, _Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits > &_-
_e, regex_constants::match_flag_type __f=regex_constants::match_default)
• template<typename _Ch_type, typename _Rx_traits >
bool std::regex_search (const _Ch_type *__s, const basic_regex< _Ch_-
type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_-
constants::match_default)
• template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename
_Rx_traits >
bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _String_-
allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_-
constants::match_flag_type __flags=regex_constants::match_default)
• template<typename _Ch_traits, typename _Ch_alloc, typename _Allocator, typename _Ch_-
type, typename _Rx_traits >
bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc
> &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _-
Ch_alloc >::const_iterator, _Allocator > &__m, const basic_regex< _-
Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_-
constants::match_default)
• template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type
>
_Out_iter std::regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __-
last, const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string<
_Ch_type > &__fmt, regex_constants::match_flag_type __flags=regex_-
constants::match_default)
• template<typename _Rx_traits, typename _Ch_type >
basic_string< _Ch_type > std::regex_replace (const basic_string< _Ch_-
type > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, const
basic_string< _Ch_type > &__fmt, regex_constants::match_flag_type _-
flags=regex_constants::match_default)

```

### 6.256.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

Definition in file [regex.h](#).

## 6.257 `regex_compiler.h` File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _InIter, typename _TraitsT>  
_AutomatonPtr std::__regex::__compile (const _InIter &__b, const _InIter &__e, _TraitsT &__t, regex_constants::syntax_option_type __f)`

#### 6.257.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

Definition in file [regex\\_compiler.h](#).

## 6.258 `regex_constants.h` File Reference

Constant definitions for the std regex library.

### Namespaces

- namespace [std](#)
- namespace [std::regex\\_constants](#)

### 5.1 Regular Expression Syntax Options

- `enum std::regex\_constants::\_\_syntax\_option {  
    _S_icase, _S_nosubs, _S_optimize, _S_collate,  
    _S_ECMAScript, _S_basic, _S_extended, _S_awk,  
    _S_grep, _S_egrep, _S_syntax_last }`
- `typedef unsigned int std::regex\_constants::syntax\_option\_type`
- `static const syntax_option_type std::regex\_constants::icase`
- `static const syntax_option_type std::regex\_constants::nosubs`
- `static const syntax_option_type std::regex\_constants::optimize`
- `static const syntax_option_type std::regex\_constants::collate`
- `static const syntax_option_type std::regex\_constants::ECMAScript`
- `static const syntax_option_type std::regex\_constants::basic`

- static const syntax\_option\_type [std::regex\\_constants::extended](#)
- static const syntax\_option\_type [std::regex\\_constants::awk](#)
- static const syntax\_option\_type [std::regex\\_constants::grep](#)
- static const syntax\_option\_type [std::regex\\_constants::egrep](#)

## 5.2 Matching Rules

Matching a regular expression against a sequence of characters [first, last) proceeds according to the rules of the grammar specified for the regular expression object, modified according to the effects listed below for any bitmask elements set.

- enum [std::regex\\_constants::\\_\\_match\\_flag](#) {  
[\\_S\\_not\\_bol](#), [\\_S\\_not\\_eol](#), [\\_S\\_not\\_bow](#), [\\_S\\_not\\_eow](#),  
[\\_S\\_any](#), [\\_S\\_not\\_null](#), [\\_S\\_continuous](#), [\\_S\\_prev\\_avail](#),  
[\\_S\\_sed](#), [\\_S\\_no\\_copy](#), [\\_S\\_first\\_only](#), [\\_S\\_match\\_flag\\_last](#) }
- typedef [std::bitset](#)< [\\_S\\_match\\_flag\\_last](#) > [std::regex\\_constants::match\\_flag\\_type](#)
- static const match\_flag\_type [std::regex\\_constants::match\\_default](#)
- static const match\_flag\_type [std::regex\\_constants::match\\_not\\_bol](#)
- static const match\_flag\_type [std::regex\\_constants::match\\_not\\_eol](#)
- static const match\_flag\_type [std::regex\\_constants::match\\_not\\_bow](#)
- static const match\_flag\_type [std::regex\\_constants::match\\_not\\_eow](#)
- static const match\_flag\_type [std::regex\\_constants::match\\_any](#)
- static const match\_flag\_type [std::regex\\_constants::match\\_not\\_null](#)
- static const match\_flag\_type [std::regex\\_constants::match\\_continuous](#)
- static const match\_flag\_type [std::regex\\_constants::match\\_prev\\_avail](#)
- static const match\_flag\_type [std::regex\\_constants::format\\_default](#)
- static const match\_flag\_type [std::regex\\_constants::format\\_sed](#)
- static const match\_flag\_type [std::regex\\_constants::format\\_no\\_copy](#)
- static const match\_flag\_type [std::regex\\_constants::format\\_first\\_only](#)

### 6.258.1 Detailed Description

Constant definitions for the std regex library. This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

Definition in file [regex\\_constants.h](#).

## 6.259 `regex_cursor.h` File Reference

### Namespaces

- namespace [std](#)



## Functions

- `template<typename _FwdIterT > _SpecializedCursor< _FwdIterT > std::__regex::__cursor (const _FwdIterT &__b, const _FwdIterT __e)`

### 6.259.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

Definition in file [regex\\_cursor.h](#).

## 6.260 `regex_error.h` File Reference

Error and exception objects for the std regex library.

## Classes

- class [std::regex\\_error](#)  
*A regular expression exception class.*  
*The regular expression library throws objects of this class on error.*

## Namespaces

- namespace [std](#)
- namespace [std::regex\\_constants](#)

## Functions

- `void std::__throw_regex_error (regex_constants::error_type __ecode)`

### 5.3 Error Types

- `enum std::regex_constants::error_type {`  
`_S_error_collate, _S_error_ctype, _S_error_escape, _S_error_backref,`  
`_S_error_brack, _S_error_paren, _S_error_brace, _S_error_badbrace,`  
`_S_error_range, _S_error_space, _S_error_badrepeat, _S_error_-`  
`complexity,`  
`_S_error_stack, _S_error_last }`

- static const error\_type `std::regex_constants::error_collate` (`_S_error_collate`)
- static const error\_type `std::regex_constants::error_ctype` (`_S_error_ctype`)
- static const error\_type `std::regex_constants::error_escape` (`_S_error_escape`)
- static const error\_type `std::regex_constants::error_backref` (`_S_error_backref`)
- static const error\_type `std::regex_constants::error_brack` (`_S_error_brack`)
- static const error\_type `std::regex_constants::error_paren` (`_S_error_paren`)
- static const error\_type `std::regex_constants::error_brace` (`_S_error_brace`)
- static const error\_type `std::regex_constants::error_badbrace` (`_S_error_badbrace`)
- static const error\_type `std::regex_constants::error_range` (`_S_error_range`)
- static const error\_type `std::regex_constants::error_space` (`_S_error_space`)
- static const error\_type `std::regex_constants::error_badrepeat` (`_S_error_badrepeat`)
- static const error\_type `std::regex_constants::error_complexity` (`_S_error_complexity`)
- static const error\_type `std::regex_constants::error_stack` (`_S_error_stack`)

#### 6.260.1 Detailed Description

Error and exception objects for the std regex library. This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

Definition in file `regex_error.h`.

## 6.261 `regex_grep_matcher.h` File Reference

### Namespaces

- namespace `std`

### Typedefs

- typedef `std::stack<_StateIdT, std::vector<_StateIdT>>` `std::__regex::StateStack`

#### 6.261.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

Definition in file `regex_grep_matcher.h`.

## 6.262 `regex_grep_matcher.tcc` File Reference

### Namespaces

- namespace [std](#)

### 6.262.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

Definition in file [regex\\_grep\\_matcher.tcc](#).

## 6.263 `regex_nfa.h` File Reference

### Namespaces

- namespace [std](#)

### Typedefs

- typedef [std::shared\\_ptr](#)< `_Automaton` > **`std::__regex::_AutomatonPtr`**
- typedef `std::function`< `bool(const _PatternCursor &)` > **`std::__regex::_Matcher`**
- typedef `int` **`std::__regex::_StateIdT`**
- typedef [std::set](#)< `_StateIdT` > **`std::__regex::_StateSet`**
- typedef `std::function`< `void(const _PatternCursor &, _Results &)` > **`std::__regex::_Tagger`**

### Enumerations

- enum `_Opcode` {  
    `_S_opcode_unknown`, `_S_opcode_alternative`, `_S_opcode_subexpr_begin`,  
    `_S_opcode_subexpr_end`,  
    `_S_opcode_match`, `_S_opcode_accept` }

### Functions

- `bool std::__regex::_AnyMatcher(const _PatternCursor &)`

## Variables

- static const `_StateIdT` `std::__regex::_S_invalid_state_id`

### 6.263.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

Definition in file [regex\\_nfa.h](#).

## 6.264 `regex_nfa.tcc` File Reference

### Namespaces

- namespace [std](#)

### 6.264.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

Definition in file [regex\\_nfa.tcc](#).

## 6.265 `rope` File Reference

### Classes

- class [\\_\\_gnu\\_cxx::rope](#)`<_CharT, _Alloc >`

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [\\_\\_gnu\\_cxx::\\_\\_detail](#)
- namespace [std](#)
- namespace [std::tr1](#)

### Defines

- `#define __GC_CONST`
- `#define __ROPE_DEFINE_ALLOC(_Tp, __name)`
- `#define __ROPE_DEFINE_ALLOC(_Tp, __name)`

- #define **\_\_ROPE\_DEFINE\_ALLOCS**(\_\_a)
- #define **\_\_STATIC\_IF\_SGI\_ALLOC**
- #define **\_\_STL\_FREE\_STRING**(\_\_s, \_\_l, \_\_a)
- #define **\_\_STL\_ROPE\_FROM\_UNOWNED\_CHAR\_PTR**(\_\_s, \_\_size, \_\_a)

## Typedefs

- typedef rope< char > **\_\_gnu\_cxx::crope**
- typedef rope< wchar\_t > **\_\_gnu\_cxx::wrope**

## Enumerations

- enum { **\_S\_max\_rope\_depth** }
- enum **\_Tag** { **\_S\_leaf**, **\_S\_concat**, **\_S\_substringfn**, **\_S\_function** }

## Functions

- crope::reference **\_\_gnu\_cxx::\_\_mutable\_reference\_at** (crope &\_\_c, size\_t \_\_i)
- template<typename \_ForwardIterator, typename \_Tp >  
void **\_\_gnu\_cxx::Destroy\_const** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, allocator< \_Tp >)
- template<typename \_ForwardIterator, typename \_Allocator >  
void **\_\_gnu\_cxx::Destroy\_const** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, \_Allocator \_\_alloc)
- template<class \_CharT >  
void **\_\_gnu\_cxx::S\_cond\_store\_eos** (\_CharT &)
- void **\_\_gnu\_cxx::S\_cond\_store\_eos** (char &\_\_c)
- void **\_\_gnu\_cxx::S\_cond\_store\_eos** (wchar\_t &\_\_c)
- template<class \_CharT >  
\_CharT **\_\_gnu\_cxx::S\_eos** (\_CharT \*)
- bool **\_\_gnu\_cxx::S\_is\_basic\_char\_type** (wchar\_t \*)
- template<class \_CharT >  
bool **\_\_gnu\_cxx::S\_is\_basic\_char\_type** (\_CharT \*)
- bool **\_\_gnu\_cxx::S\_is\_basic\_char\_type** (char \*)
- template<class \_CharT >  
bool **\_\_gnu\_cxx::S\_is\_one\_byte\_char\_type** (\_CharT \*)
- bool **\_\_gnu\_cxx::S\_is\_one\_byte\_char\_type** (char \*)
- template<class \_CharT, class \_Alloc >  
bool **\_\_gnu\_cxx::operator!=** (const \_Rope\_iterator< \_CharT, \_Alloc > &\_\_x, const \_Rope\_iterator< \_CharT, \_Alloc > &\_\_y)

- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator!= (const rope< _CharT, _Alloc > &__x, const`  
`rope< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator!= (const _Rope_char_ptr_proxy< _CharT, _Alloc`  
`> &__x, const _Rope_char_ptr_proxy< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator!= (const _Rope_const_iterator< _CharT, _Alloc >`  
`&__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`_Rope_iterator< _CharT, _Alloc > __gnu_cxx::operator+ (const _Rope_-`  
`iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >`  
`_Rope_iterator< _CharT, _Alloc > __gnu_cxx::operator+ (ptrdiff_t __n, const`  
`_Rope_iterator< _CharT, _Alloc > &__x)`
- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > __gnu_cxx::operator+ (const rope< _CharT, _Alloc`  
`> &__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > __gnu_cxx::operator+ (const rope< _CharT, _Alloc`  
`> &__left, const _CharT *__right)`
- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > __gnu_cxx::operator+ (const rope< _CharT, _Alloc`  
`> &__left, _CharT __right)`
- `template<class _CharT, class _Alloc >`  
`_Rope_const_iterator< _CharT, _Alloc > __gnu_cxx::operator+ (const _-`  
`Rope_const_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >`  
`_Rope_const_iterator< _CharT, _Alloc > __gnu_cxx::operator+ (ptrdiff_t __-`  
`n, const _Rope_const_iterator< _CharT, _Alloc > &__x)`
- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > & __gnu_cxx::operator+= (rope< _CharT, _Alloc >`  
`&__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > & __gnu_cxx::operator+= (rope< _CharT, _Alloc >`  
`&__left, const _CharT *__right)`
- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > & __gnu_cxx::operator+= (rope< _CharT, _Alloc >`  
`&__left, _CharT __right)`
- `template<class _CharT, class _Alloc >`  
`ptrdiff_t __gnu_cxx::operator- (const _Rope_const_iterator< _CharT, _Alloc`  
`> &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`_Rope_iterator< _CharT, _Alloc > __gnu_cxx::operator- (const _Rope_-`  
`iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`

- `template<class _CharT, class _Alloc >`  
`ptrdiff_t __gnu_cxx::operator- (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`_Rope_const_iterator< _CharT, _Alloc > __gnu_cxx::operator- (const _Rope_const_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator< (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator< (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator< (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Traits, class _Alloc >`  
`std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &__o, const rope< _CharT, _Alloc > &__r)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator<= (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator<= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator<= (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator== (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator== (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator== (const _Rope_char_ptr_proxy< _CharT, _Alloc > &__x, const _Rope_char_ptr_proxy< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator== (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator> (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`

- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator> (const _Rope_iterator< _CharT, _Alloc > &__x,`  
`const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator> (const _Rope_const_iterator< _CharT, _Alloc >`  
`&__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator>= (const _Rope_const_iterator< _CharT, _Alloc >`  
`&__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator>= (const _Rope_iterator< _CharT, _Alloc > &__x,`  
`const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator>= (const rope< _CharT, _Alloc > &__x, const`  
`rope< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`void __gnu_cxx::swap (_Rope_char_ref_proxy< _CharT, _Alloc > __a, _`  
`Rope_char_ref_proxy< _CharT, _Alloc > __b)`
- `template<class _CharT, class _Alloc >`  
`void __gnu_cxx::swap (rope< _CharT, _Alloc > &__x, rope< _CharT, _Alloc`  
`> &__y)`

## Variables

- `rope< _CharT, _Alloc > __gnu_cxx::identity_element (_Rope_Concat_fn<`  
`_CharT, _Alloc >)`

### 6.265.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [rope](#).

## 6.266 ropeimpl.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Functions

- `template<class _CharT, class _Traits >`



```

void __gnu_cxx::_Rope_fill (basic_ostream< _CharT, _Traits > &__o, size_t
__n)
• template<class _CharT >
 bool __gnu_cxx::_Rope_is_simple (_CharT *)
• bool __gnu_cxx::_Rope_is_simple (wchar_t *)
• bool __gnu_cxx::_Rope_is_simple (char *)
• template<class _Rope_iterator >
 void __gnu_cxx::_Rope_rotate (_Rope_iterator __first, _Rope_iterator __-
 middle, _Rope_iterator __last)
• template<class _CharT, class _Traits, class _Alloc >
 std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<<
 (std::basic_ostream< _CharT, _Traits > &__o, const rope< _CharT, _Alloc >
 &__r)
• void __gnu_cxx::rotate (_Rope_iterator< char, __STL_DEFAULT_-
 ALLOCATOR(char)> __first, _Rope_iterator< char, __STL_DEFAULT_-
 ALLOCATOR(char)> __middle, _Rope_iterator< char, __STL_DEFAULT_-
 ALLOCATOR(char)> __last)

```

### 6.266.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/rope>`.

Definition in file [ropeimpl.h](#).

## 6.267 `safe_base.h` File Reference

### Classes

- class [\\_\\_gnu\\_debug::\\_Safe\\_iterator\\_base](#)  
*Basic functionality for a safe iterator.*
- class [\\_\\_gnu\\_debug::\\_Safe\\_sequence\\_base](#)  
*Base class that supports tracking of iterators that reference a sequence.*

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)

### 6.267.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [safe\\_base.h](#).

## 6.268 `safe_iterator.h` File Reference

### Classes

- struct [\\_\\_gnu\\_debug::BeforeBeginHelper<\\_Sequence>](#)
- class [\\_\\_gnu\\_debug::Safe\\_iterator<\\_Iterator, \\_Sequence>](#)  
*Safe iterator wrapper.*

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)

### Functions

- template<typename \_Iterator >  
  [\\_Siter\\_base<\\_Iterator>::iterator\\_type](#) [\\_\\_gnu\\_debug::\\_\\_base](#) (\_Iterator \_\_it)
- bool [\\_\\_gnu\\_debug::\\_\\_check\\_singular\\_aux](#) (const [\\_Safe\\_iterator\\_base](#) \*\_\_x)
- template<typename \_IteratorL, typename \_IteratorR, typename \_Sequence >  
  bool [\\_\\_gnu\\_debug::operator!=](#) (const [\\_Safe\\_iterator<\\_IteratorL, \\_Sequence>](#) &\_\_lhs, const [\\_Safe\\_iterator<\\_IteratorR, \\_Sequence>](#) &\_\_rhs)
- template<typename \_Iterator, typename \_Sequence >  
  bool [\\_\\_gnu\\_debug::operator!=](#) (const [\\_Safe\\_iterator<\\_Iterator, \\_Sequence>](#) &\_\_lhs, const [\\_Safe\\_iterator<\\_Iterator, \\_Sequence>](#) &\_\_rhs)
- template<typename \_Iterator, typename \_Sequence >  
  [\\_Safe\\_iterator<\\_Iterator, \\_Sequence>](#) [\\_\\_gnu\\_debug::operator+](#) (typename [\\_Safe\\_iterator<\\_Iterator, \\_Sequence>](#)::difference\_type \_\_n, const [\\_Safe\\_iterator<\\_Iterator, \\_Sequence>](#) &\_\_i)
- template<typename \_IteratorL, typename \_IteratorR, typename \_Sequence >  
  typename [\\_Safe\\_iterator<\\_IteratorL, \\_Sequence>](#)::difference\_type [\\_\\_gnu\\_debug::operator-](#) (const [\\_Safe\\_iterator<\\_IteratorL, \\_Sequence>](#) &\_\_lhs, const [\\_Safe\\_iterator<\\_IteratorR, \\_Sequence>](#) &\_\_rhs)
- template<typename \_Iterator, typename \_Sequence >  
  typename [\\_Safe\\_iterator<\\_Iterator, \\_Sequence>](#)::difference\_type [\\_\\_gnu\\_debug::operator-](#) (const [\\_Safe\\_iterator<\\_Iterator, \\_Sequence>](#) &\_\_lhs, const [\\_Safe\\_iterator<\\_Iterator, \\_Sequence>](#) &\_\_rhs)

- `template<typename _Iterator, typename _Sequence >`  
`bool __gnu_debug::operator< (const _Safe_iterator< _Iterator, _Sequence >`  
`&__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool __gnu_debug::operator< (const _Safe_iterator< _IteratorL, _Sequence`  
`> &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool __gnu_debug::operator<= (const _Safe_iterator< _IteratorL, _Sequence`  
`> &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool __gnu_debug::operator<= (const _Safe_iterator< _Iterator, _Sequence`  
`> &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool __gnu_debug::operator== (const _Safe_iterator< _IteratorL, _Sequence`  
`> &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool __gnu_debug::operator== (const _Safe_iterator< _Iterator, _Sequence`  
`&__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool __gnu_debug::operator> (const _Safe_iterator< _Iterator, _Sequence`  
`&__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool __gnu_debug::operator> (const _Safe_iterator< _IteratorL, _Sequence`  
`> &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool __gnu_debug::operator>= (const _Safe_iterator< _IteratorL, _Sequence`  
`> &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool __gnu_debug::operator>= (const _Safe_iterator< _Iterator, _Sequence`  
`> &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)`

### 6.268.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [safe\\_iterator.h](#).

## 6.269 `safe_iterator.tcc` File Reference

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)

### 6.269.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [safe\\_iterator.tcc](#).

## 6.270 `safe_sequence.h` File Reference

### Classes

- class [\\_\\_gnu\\_debug::\\_After\\_nth\\_from<\\_Iterator>](#)
- class [\\_\\_gnu\\_debug::\\_Equal\\_to<\\_Type>](#)
- class [\\_\\_gnu\\_debug::\\_Not\\_equal\\_to<\\_Type>](#)
- class [\\_\\_gnu\\_debug::\\_Safe\\_sequence<\\_Sequence>](#)

*Base class for constructing a safe sequence type that tracks iterators that reference it.*

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)

### 6.270.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [safe\\_sequence.h](#).

## 6.271 `safe_sequence.tcc` File Reference

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)

### 6.271.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [safe\\_sequence.tcc](#).

## 6.272 `search.h` File Reference

Parallel implementation base for `std::search()` and `std::search_n()`. This file is a GNU parallel extension to the Standard C++ Library.

## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

## Functions

- `template<typename _RAIter, typename _DifferenceTp >  
void \_\_gnu\_parallel::\_\_calc\_borders (_RAIter __elements, _DifferenceTp __length, _DifferenceTp *__off)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred >  
__RAIter1 \_\_gnu\_parallel::\_\_search\_template (__RAIter1 __begin1, __RAIter1 __end1, __RAIter2 __begin2, __RAIter2 __end2, _Pred __pred)`

### 6.272.1 Detailed Description

Parallel implementation base for `std::search()` and `std::search_n()`. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [search.h](#).

## 6.273 set File Reference

### 6.273.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [set](#).

## 6.274 set File Reference

### 6.274.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/set](#).

## 6.275 set File Reference

### 6.275.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/set](#).

## 6.276 set.h File Reference

### Classes

- class [std::\\_\\_debug::set<\\_Key, \\_Compare, \\_Allocator>](#)  
*Class [std::set](#) with safety/checking/debug instrumentation.*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_debug](#)

### Functions

- `template<typename _Key, typename _Compare, typename _Allocator>  
bool std::__debug::operator!= (const set< _Key, _Compare, _Allocator > &_  
_lhs, const set< _Key, _Compare, _Allocator > &_rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>  
bool std::__debug::operator< (const set< _Key, _Compare, _Allocator > &_  
_lhs, const set< _Key, _Compare, _Allocator > &_rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>  
bool std::__debug::operator<= (const set< _Key, _Compare, _Allocator > &_  
_lhs, const set< _Key, _Compare, _Allocator > &_rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>  
bool std::__debug::operator== (const set< _Key, _Compare, _Allocator > &_  
_lhs, const set< _Key, _Compare, _Allocator > &_rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>  
bool std::__debug::operator> (const set< _Key, _Compare, _Allocator > &_  
_lhs, const set< _Key, _Compare, _Allocator > &_rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>  
bool std::__debug::operator>= (const set< _Key, _Compare, _Allocator > &_  
_lhs, const set< _Key, _Compare, _Allocator > &_rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>  
void std::__debug::swap (set< _Key, _Compare, _Allocator > &_x, set< _  
Key, _Compare, _Allocator > &_y)`

#### 6.276.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/set.h](#).

## 6.277 set.h File Reference

### Classes

- class [std::\\_\\_profile::set< \\_Key, \\_Compare, \\_Allocator >](#)  
*Class [std::set](#) wrapper with performance instrumentation.*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_profile](#)

### Functions

- `template<typename _Key, typename _Compare, typename _Allocator >  
bool std::__profile::operator!= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >  
bool std::__profile::operator< (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >  
bool std::__profile::operator<= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >  
bool std::__profile::operator== (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >  
bool std::__profile::operator> (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >  
bool std::__profile::operator>= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >  
void std::__profile::swap (set< _Key, _Compare, _Allocator > &__x, set< _Key, _Compare, _Allocator > &__y)`

#### 6.277.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/set.h](#).

## 6.278 `set_operations.h` File Reference

Parallel implementations of set operations for random-access iterators. This file is a GNU parallel extension to the Standard C++ Library.

### Namespaces

- namespace `__gnu_parallel`

### Functions

- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator __gnu_parallel::__copy_tail (std::pair< _Iter, _Iter > __b,`  
`std::pair< _Iter, _Iter > __e, _OutputIterator __r)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`  
`_OutputIterator __gnu_parallel::__parallel_set_difference (_Iter __begin1,`  
`_Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _`  
`Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`  
`_OutputIterator __gnu_parallel::__parallel_set_intersection (_Iter __begin1,`  
`_Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _`  
`Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Operation >`  
`_OutputIterator __gnu_parallel::__parallel_set_operation (_Iter __begin1,`  
`_Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _`  
`Operation __op)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`  
`_OutputIterator __gnu_parallel::__parallel_set_symmetric_difference (_Iter`  
`__begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __`  
`result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`  
`_OutputIterator __gnu_parallel::__parallel_set_union (_Iter __begin1, _Iter`  
`__end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _Compare __`  
`comp)`

#### 6.278.1 Detailed Description

Parallel implementations of set operations for random-access iterators. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file `set_operations.h`.



## 6.279 settings.h File Reference

Runtime settings and tuning parameters, heuristics to decide whether to use parallelized algorithms. This file is a GNU parallel extension to the Standard C++ Library.

### Classes

- struct [\\_\\_gnu\\_parallel::\\_Settings](#)  
*class [\\_Settings](#) /// Run-time settings for the parallel mode including all tunable parameters.*

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Defines

- #define [\\_GLIBCXX\\_PARALLEL\\_CONDITION](#)(\_\_c)

#### 6.279.1 Detailed Description

Runtime settings and tuning parameters, heuristics to decide whether to use parallelized algorithms. This file is a GNU parallel extension to the Standard C++ Library.

#### 6.279.2 parallelization\_decision

The decision whether to run an algorithm in parallel.

There are several ways the user can switch on and \_\_off the parallel execution of an algorithm, both at compile- and run-time.

Only sequential execution can be forced at compile-time. This reduces code size and protects code parts that have non-thread-safe side effects.

Ultimately, forcing parallel execution at compile-time makes sense. Often, the sequential algorithm implementation is used as a subroutine, so no reduction in code size can be achieved. Also, the machine the program is run on might have only one processor core, so to avoid overhead, the algorithm is executed sequentially.

To force sequential execution of an algorithm ultimately at compile-time, the user must add the tag `gnu_parallel::sequential_tag()` to the end of the parameter list, e. g.

```
std::sort(__v.begin(), __v.end(), __gnu_parallel::sequential_tag());
```

This is compatible with all overloaded algorithm variants. No additional code will be instantiated, at all. The same holds for most algorithm calls with iterators not providing random access.

If the algorithm call is not forced to be executed sequentially at compile-time, the decision is made at run-time. The global variable `__gnu_parallel::_Settings::algorithm_strategy` is checked. `_It` is a tristate variable corresponding to:

a. `force_sequential`, meaning the sequential algorithm is executed. b. `force_parallel`, meaning the parallel algorithm is executed. c. `heuristic`

For `heuristic`, the parallel algorithm implementation is called only if the input size is sufficiently large. For most algorithms, the input size is the (combined) length of the input sequence(`__s`). The threshold can be set by the user, individually for each algorithm. The according variables are called `gnu_parallel::_Settings::[algorithm]_minimal_n`.

For some of the algorithms, there are even more tuning options, e. g. the ability to choose from multiple algorithm variants. See below for details.

Definition in file [settings.h](#).

### 6.279.3 Define Documentation

#### 6.279.3.1 `#define _GLIBCXX_PARALLEL_CONDITION( __c )`

Determine at compile(?)-time if the parallel variant of an algorithm should be called.

#### Parameters

`__c` A condition that is convertible to `bool` that is overruled by `__gnu_parallel::_Settings::algorithm_strategy`. Usually a decision based on the input size.

Definition at line 95 of file `settings.h`.

## 6.280 `shared_ptr.h` File Reference

### Classes

- class [std::enable\\_shared\\_from\\_this<\\_Tp>](#)  
*Base class allowing use of member function `shared_from_this`.*
- struct [std::hash<shared\\_ptr<\\_Tp>>](#)  
*`std::hash` specialization for `shared_ptr`.*

- struct `std::owner_less< shared_ptr< _Tp > >`  
*Partial specialization of `owner_less` for `shared_ptr`.*
- struct `std::owner_less< weak_ptr< _Tp > >`  
*Partial specialization of `owner_less` for `weak_ptr`.*
- class `std::shared_ptr< _Tp >`  
*A smart pointer with reference-counted copy semantics.*
- class `std::weak_ptr< _Tp >`  
*A smart pointer with weak semantics.*

## Namespaces

- namespace `std`

## Functions

- template<typename \_Tp, typename \_Alloc, typename... \_Args>  
`shared_ptr< _Tp >` `std::allocate_shared` (const \_Alloc &\_\_a, \_Args &&...\_\_args)
- template<typename \_Tp, typename \_Tp1 >  
`shared_ptr< _Tp >` `std::const_pointer_cast` (const `shared_ptr< _Tp1 >` &\_\_r)
- template<typename \_Tp, typename \_Tp1 >  
`shared_ptr< _Tp >` `std::dynamic_pointer_cast` (const `shared_ptr< _Tp1 >` &\_\_r)
- template<typename \_Del, typename \_Tp, \_Lock\_policy \_Lp>  
`_Del *` `std::get_deleter` (const `__shared_ptr< _Tp, _Lp >` &\_\_p)
- template<typename \_Tp, typename... \_Args>  
`shared_ptr< _Tp >` `std::make_shared` (\_Args &&...\_\_args)
- template<typename \_Tp >  
bool `std::operator!=` (nullptr\_t, const `shared_ptr< _Tp >` &\_\_b)
- template<typename \_Tp1, typename \_Tp2 >  
bool `std::operator!=` (const `shared_ptr< _Tp1 >` &\_\_a, const `shared_ptr< _Tp2 >` &\_\_b)
- template<typename \_Tp >  
bool `std::operator!=` (const `shared_ptr< _Tp >` &\_\_a, nullptr\_t)
- template<typename \_Tp1, typename \_Tp2 >  
bool `std::operator<` (const `shared_ptr< _Tp1 >` &\_\_a, const `shared_ptr< _Tp2 >` &\_\_b)

- `template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp>`  
`std::basic_ostream< _Ch, _Tr > & std::operator<< (std::basic_ostream< _Ch, _Tr > &__os, const __shared_ptr< _Tp, _Lp > &__p)`
- `template<typename _Tp >`  
`bool std::operator== (const shared_ptr< _Tp > &__a, nullptr_t)`
- `template<typename _Tp >`  
`bool std::operator== (nullptr_t, const shared_ptr< _Tp > &__b)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool std::operator== (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b)`
- `template<typename _Tp, typename _Tp1 >`  
`shared_ptr< _Tp > std::static_pointer_cast (const shared_ptr< _Tp1 > &__r)`
- `template<typename _Tp >`  
`void std::swap (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b)`
- `template<typename _Tp >`  
`void std::swap (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b)`

### 6.280.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

Definition in file `shared_ptr.h`.

## 6.281 `shared_ptr_base.h` File Reference

### Classes

- class `std::bad_weak_ptr`  
*Exception possibly thrown by `shared_ptr`.*
- struct `std::hash< __shared_ptr< _Tp, _Lp > >`  
*`std::hash` specialization for `__shared_ptr`.*

### Namespaces

- namespace `std`

## Functions

- `template<typename _Tp, _Lock_policy _Lp, typename _Alloc, typename... _Args>  
__shared_ptr< _Tp, _Lp > std::__allocate_shared (const _Alloc &__a, _Args  
&&...__args)`
- `template<_Lock_policy _Lp, typename _Tp1, typename _Tp2 >  
void std::__enable_shared_from_this_helper (const __shared_count< _Lp >  
&, const __enable_shared_from_this< _Tp1, _Lp > *, const _Tp2 *)`
- `template<_Lock_policy _Lp>  
void std::__enable_shared_from_this_helper (const __shared_count< _Lp >  
&,...)`
- `template<typename _Tp1, typename _Tp2 >  
void std::__enable_shared_from_this_helper (const __shared_count<> &  
const enable_shared_from_this< _Tp1 > *, const _Tp2 *)`
- `template<typename _Tp, _Lock_policy _Lp, typename... _Args>  
__shared_ptr< _Tp, _Lp > std::__make_shared (_Args &&...__args)`
- `void std::__throw_bad_weak_ptr ()`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>  
__shared_ptr< _Tp, _Lp > std::const_pointer_cast (const __shared_ptr< _Tp1,  
_Lp > &__r)`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>  
__shared_ptr< _Tp, _Lp > std::dynamic_pointer_cast (const __shared_ptr< _  
Tp1, _Lp > &__r)`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>  
bool std::operator!= (const __shared_ptr< _Tp1, _Lp > &__a, const __-  
shared_ptr< _Tp2, _Lp > &__b)`
- `template<typename _Tp, _Lock_policy _Lp>  
bool std::operator!= (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t)`
- `template<typename _Tp, _Lock_policy _Lp>  
bool std::operator!= (nullptr_t, const __shared_ptr< _Tp, _Lp > &__b)`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>  
bool std::operator< (const __shared_ptr< _Tp1, _Lp > &__a, const __-  
shared_ptr< _Tp2, _Lp > &__b)`
- `template<typename _Tp, _Lock_policy _Lp>  
bool std::operator== (nullptr_t, const __shared_ptr< _Tp, _Lp > &__b)`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>  
bool std::operator== (const __shared_ptr< _Tp1, _Lp > &__a, const __-  
shared_ptr< _Tp2, _Lp > &__b)`
- `template<typename _Tp, _Lock_policy _Lp>  
bool std::operator== (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t)`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>  
__shared_ptr< _Tp, _Lp > std::static_pointer_cast (const __shared_ptr< _Tp1,  
_Lp > &__r)`

- `template<typename _Tp, _Lock_policy _Lp>`  
`void std::swap (__shared_ptr< _Tp, _Lp > &__a, __shared_ptr< _Tp, _Lp >`  
`&__b)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`void std::swap (__weak_ptr< _Tp, _Lp > &__a, __weak_ptr< _Tp, _Lp >`  
`&__b)`

### 6.281.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

Definition in file [shared\\_ptr\\_base.h](#).

## 6.282 `slice_array.h` File Reference

### Classes

- class [std::slice](#)  
*Class defining one-dimensional subset of an array.*
- class [std::slice\\_array< \\_Tp >](#)  
*Reference to one-dimensional subset of an array.*

### Namespaces

- namespace [std](#)

### Defines

- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`

### 6.282.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

Definition in file [slice\\_array.h](#).

## 6.283 slist File Reference

### Classes

- class [\\_\\_gnu\\_cxx::slist< \\_Tp, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

### Functions

- `_Slist_node_base * __gnu_cxx::__slist_make_link (_Slist_node_base * __prev_node, _Slist_node_base * __new_node)`
- `_Slist_node_base * __gnu_cxx::__slist_previous (_Slist_node_base * __head, const _Slist_node_base * __node)`
- `const _Slist_node_base * __gnu_cxx::__slist_previous (const _Slist_node_base * __head, const _Slist_node_base * __node)`
- `_Slist_node_base * __gnu_cxx::__slist_reverse (_Slist_node_base * __node)`
- `size_t __gnu_cxx::__slist_size (_Slist_node_base * __node)`
- `void __gnu_cxx::__slist_splice_after (_Slist_node_base * __pos, _Slist_node_base * __before_first, _Slist_node_base * __before_last)`
- `void __gnu_cxx::__slist_splice_after (_Slist_node_base * __pos, _Slist_node_base * __head)`
- `template<class _Tp, class _Alloc >  
bool __gnu_cxx::operator!= (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)`
- `template<class _Tp, class _Alloc >  
bool __gnu_cxx::operator< (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)`
- `template<class _Tp, class _Alloc >  
bool __gnu_cxx::operator<= (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)`
- `template<class _Tp, class _Alloc >  
bool __gnu_cxx::operator== (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)`
- `template<class _Tp, class _Alloc >  
bool __gnu_cxx::operator> (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)`
- `template<class _Tp, class _Alloc >  
bool __gnu_cxx::operator>= (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)`

- `template<class _Tp, class _Alloc >`  
`void __gnu_cxx::swap (slist< _Tp, _Alloc > &__x, slist< _Tp, _Alloc > &__y)`

### 6.283.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [slist](#).

## 6.284 **sort.h** File Reference

Parallel sorting algorithm switch. This file is a GNU parallel extension to the Standard C++ Library.

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- `template<bool __stable, typename _RAIter, typename _Compare, typename _Parallelism >`  
`void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, parallel_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, default_parallel_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, balanced_quicksort_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, quicksort_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_sampling_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_exact_tag __parallelism)`



- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _-`  
`Compare __comp, multiway_mergesort_tag __parallelism)`

#### 6.284.1 Detailed Description

Parallel sorting algorithm switch. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [sort.h](#).

### 6.285 sso\_string\_base.h File Reference

#### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

#### 6.285.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/vstring.h>`.

Definition in file [sso\\_string\\_base.h](#).

### 6.286 sstream File Reference

#### Classes

- class [std::basic\\_istream<\\_CharT, \\_Traits, \\_Alloc >](#)  
*Controlling input for std::string.*  
*This class supports reading from objects of type [std::basic\\_string](#), using the inherited functions from [std::basic\\_istream](#). To control the associated sequence, an instance of [std::basic\\_stringbuf](#) is used, which this page refers to as *sb*.*
- class [std::basic\\_ostringstream<\\_CharT, \\_Traits, \\_Alloc >](#)  
*Controlling output for std::string.*  
*This class supports writing to objects of type [std::basic\\_string](#), using the inherited functions from [std::basic\\_ostream](#). To control the associated sequence, an instance of [std::basic\\_stringbuf](#) is used, which this page refers to as *sb*.*
- class [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc >](#)  
*The actual work of input and output (for std::string).*

*This class associates either or both of its input and output sequences with a sequence of characters, which can be initialized from, or made available as, a `std::basic_string`. (Paraphrased from [27.7.1]/1.).*

- class `std::basic_stringstream<_CharT, _Traits, _Alloc>`

*Controlling input and output for `std::string`.*

*This class supports reading from and writing to objects of type `std::basic_string`, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.*

## Namespaces

- namespace `std`

### 6.286.1 Detailed Description

This is a Standard C++ Library header.

Definition in file `sstream`.

## 6.287 `sstream.tcc` File Reference

## Namespaces

- namespace `std`

### 6.287.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<sstream>`.

Definition in file `sstream.tcc`.

## 6.288 `stack` File Reference

### 6.288.1 Detailed Description

This is a Standard C++ Library header.

Definition in file `stack`.

## 6.289 `standard_policies.hpp` File Reference

### Namespaces

- namespace `__gnu_pbds`

### Enumerations

- enum { `default_store_hash` }

#### 6.289.1 Detailed Description

Contains standard policies for containers.

Definition in file `standard_policies.hpp`.

## 6.290 `stdc++.h` File Reference

#### 6.290.1 Detailed Description

This is an implementation file for a precompiled header.

Definition in file `stdc++.h`.

## 6.291 `stdexcept` File Reference

### Classes

- class `std::domain_error`
- class `std::invalid_argument`
- class `std::length_error`
- class `std::logic_error`

*One of two subclasses of exception.*

- class `std::out_of_range`
- class `std::overflow_error`
- class `std::range_error`
- class `std::runtime_error`

*One of two subclasses of exception.*

- class `std::underflow_error`

## Namespaces

- namespace `std`

### 6.291.1 Detailed Description

This is a Standard C++ Library header.

Definition in file `stdexcept`.

## 6.292 `stdio_filebuf.h` File Reference

### Classes

- class `__gnu_cxx::stdio_filebuf<_CharT, _Traits>`  
*Provides a layer of compatibility for C/POSIX.  
This GNU extension provides extensions for working with standard C FILE\*'s and POSIX file descriptors. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.*

## Namespaces

- namespace `__gnu_cxx`

### 6.292.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file `stdio_filebuf.h`.

## 6.293 `stdio_sync_filebuf.h` File Reference

### Classes

- class `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`  
*Provides a layer of compatibility for C.  
This GNU extension provides extensions for working with standard C FILE\*'s. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.*

## Namespaces

- namespace `__gnu_cxx`

### 6.293.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file `stdio_sync_filebuf.h`.

## 6.294 `stdtr1c++.h` File Reference

### 6.294.1 Detailed Description

This is an implementation file for a precompiled header.

Definition in file `stdtr1c++.h`.

## 6.295 `stl_algo.h` File Reference

## Namespaces

- namespace `std`

## Enumerations

- enum { `_S_threshold` }
- enum { `_S_chunk_size` }

## Functions

- `template<typename _RandomAccessIterator, typename _Distance >`  
`void std::__chunk_insertion_sort` (`_RandomAccessIterator __first`, `_RandomAccessIterator __last`, `_Distance __chunk_size`)
- `template<typename _RandomAccessIterator, typename _Distance, typename _Compare >`  
`void std::__chunk_insertion_sort` (`_RandomAccessIterator __first`, `_RandomAccessIterator __last`, `_Distance __chunk_size`, `_Compare __comp`)
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`  
`_OutputIterator std::__copy_n` (`_InputIterator __first`, `_Size __n`, `_OutputIterator __result`, `input_iterator_tag`)

- `template<typename _RandomAccessIterator, typename _Size, typename _OutputIterator >`  
`_OutputIterator std::copy_n (_RandomAccessIterator __first, _Size __n, _`  
`OutputIterator __result, random_access_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::final_insertion_sort (_RandomAccessIterator __first, _`  
`RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::final_insertion_sort (_RandomAccessIterator __first, _`  
`RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Tp >`  
`_RandomAccessIterator std::find (_RandomAccessIterator __first, _`  
`RandomAccessIterator __last, const _Tp &__val, random_access_iterator_tag)`
- `template<typename _InputIterator, typename _Tp >`  
`_InputIterator std::find (_InputIterator __first, _InputIterator __last, const _Tp`  
`&__val, input_iterator_tag)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _`  
`BinaryPredicate >`  
`_BidirectionalIterator1 std::find_end (_BidirectionalIterator1 __`  
`first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2,`  
`_BidirectionalIterator2 __last2, bidirectional_iterator_tag, bidirectional_`  
`iterator_tag, _BinaryPredicate __comp)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`_ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _`  
`ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2`  
`__last2, forward_iterator_tag, forward_iterator_tag)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate`  
`>`  
`_ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _`  
`ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2`  
`__last2, forward_iterator_tag, forward_iterator_tag, _BinaryPredicate __comp)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2 >`  
`_BidirectionalIterator1 std::find_end (_BidirectionalIterator1 __`  
`first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2,`  
`_BidirectionalIterator2 __last2, bidirectional_iterator_tag, bidirectional_`  
`iterator_tag)`
- `template<typename _InputIterator, typename _Predicate >`  
`_InputIterator std::find_if (_InputIterator __first, _InputIterator __last, _`  
`Predicate __pred, input_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Predicate >`  
`_RandomAccessIterator std::find_if (_RandomAccessIterator __first, _`  
`RandomAccessIterator __last, _Predicate __pred, random_access_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Predicate >`  
`_RandomAccessIterator std::find_if_not (_RandomAccessIterator __first, _`  
`RandomAccessIterator __last, _Predicate __pred, random_access_iterator_tag)`

- `template<typename _InputIterator, typename _Predicate >`  
`_InputIterator std::__find_if_not` (`_InputIterator __first`, `_InputIterator __last`, `_`  
`Predicate __pred`, `input_iterator_tag`)
- `template<typename _EuclideanRingElement >`  
`_EuclideanRingElement std::__gcd` (`_EuclideanRingElement __m`, `_`  
`EuclideanRingElement __n`)
- `template<typename _RandomAccessIterator >`  
`void std::__heap_select` (`_RandomAccessIterator __first`, `_`  
`RandomAccessIterator __middle`, `_RandomAccessIterator __last`)
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::__heap_select` (`_RandomAccessIterator __first`, `_`  
`RandomAccessIterator __middle`, `_RandomAccessIterator __last`, `_Compare`  
`__comp`)
- `template<typename _ForwardIterator, typename _Predicate, typename _Distance >`  
`_ForwardIterator std::__inplace_stable_partition` (`_ForwardIterator __first`, `_`  
`ForwardIterator __last`, `_Predicate __pred`, `_Distance __len`)
- `template<typename _RandomAccessIterator >`  
`void std::__inplace_stable_sort` (`_RandomAccessIterator __first`, `_`  
`RandomAccessIterator __last`)
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::__inplace_stable_sort` (`_RandomAccessIterator __first`, `_`  
`RandomAccessIterator __last`, `_Compare __comp`)
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::__insertion_sort` (`_RandomAccessIterator __first`, `_`  
`RandomAccessIterator __last`, `_Compare __comp`)
- `template<typename _RandomAccessIterator >`  
`void std::__insertion_sort` (`_RandomAccessIterator __first`, `_`  
`RandomAccessIterator __last`)
- `template<typename _RandomAccessIterator, typename _Size >`  
`void std::__introsselect` (`_RandomAccessIterator __first`, `_`  
`RandomAccessIterator __nth`, `_RandomAccessIterator __last`, `_Size __`  
`depth_limit`)
- `template<typename _RandomAccessIterator, typename _Size, typename _Compare >`  
`void std::__introsselect` (`_RandomAccessIterator __first`, `_`  
`RandomAccessIterator __nth`, `_RandomAccessIterator __last`, `_Size __`  
`depth_limit`, `_Compare __comp`)
- `template<typename _RandomAccessIterator, typename _Size >`  
`void std::__introsort_loop` (`_RandomAccessIterator __first`, `_`  
`RandomAccessIterator __last`, `_Size __depth_limit`)
- `template<typename _RandomAccessIterator, typename _Size, typename _Compare >`  
`void std::__introsort_loop` (`_RandomAccessIterator __first`, `_`  
`RandomAccessIterator __last`, `_Size __depth_limit`, `_Compare __comp`)
- `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer >`  
`void std::__merge_adaptive` (`_BidirectionalIterator __first`, `_`  
`BidirectionalIterator __middle`, `_BidirectionalIterator __last`, `_Distance`  
`__len1`, `_Distance __len2`, `_Pointer __buffer`, `_Distance __buffer_size`)

- `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer, typename _Compare >`  
`void std::__merge_adaptive (_BidirectionalIterator __first, _-`  
`BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1,`  
`_Distance __len2, _Pointer __buffer, _Distance __buffer_size, _Compare`  
`__comp)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _-`  
`BidirectionalIterator3 >`  
`_BidirectionalIterator3 std::__merge_backward (_BidirectionalIterator1 __-`  
`first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _-`  
`BidirectionalIterator2 __last2, _BidirectionalIterator3 __result)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _-`  
`BidirectionalIterator3, typename _Compare >`  
`_BidirectionalIterator3 std::__merge_backward (_BidirectionalIterator1 __-`  
`first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _-`  
`BidirectionalIterator2 __last2, _BidirectionalIterator3 __result, _Compare __-`  
`comp)`
- `template<typename _RandomAccessIterator1, typename _RandomAccessIterator2, typename`  
`_Distance >`  
`void std::__merge_sort_loop (_RandomAccessIterator1 __first, _-`  
`RandomAccessIterator1 __last, _RandomAccessIterator2 __result, _Distance`  
`__step_size)`
- `template<typename _RandomAccessIterator1, typename _RandomAccessIterator2, typename`  
`_Distance, typename _Compare >`  
`void std::__merge_sort_loop (_RandomAccessIterator1 __first, _-`  
`RandomAccessIterator1 __last, _RandomAccessIterator2 __result, _Distance`  
`__step_size, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Pointer >`  
`void std::__merge_sort_with_buffer (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _Pointer __buffer)`
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Compare >`  
`void std::__merge_sort_with_buffer (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _Pointer __buffer, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Distance >`  
`void std::__merge_without_buffer (_BidirectionalIterator __first, _-`  
`BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance`  
`__len1, _Distance __len2)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Compare >`  
`void std::__merge_without_buffer (_BidirectionalIterator __first, _-`  
`BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance`  
`__len1, _Distance __len2, _Compare __comp)`
- `template<typename _Iterator, typename _Compare >`  
`void std::__move_median_first (_Iterator __a, _Iterator __b, _Iterator __c, _-`  
`Compare __comp)`



- `template<typename _Iterator >`  
`void std::__move_median_first (_Iterator __a, _Iterator __b, _Iterator __c)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator std::__partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, forward_iterator_tag)`
- `template<typename _BidirectionalIterator, typename _Predicate >`  
`_BidirectionalIterator std::__partition (_BidirectionalIterator __first, _BidirectionalIterator __last, _Predicate __pred, bidirectional_iterator_tag)`
- `template<typename _BidirectionalIterator >`  
`void std::__reverse (_BidirectionalIterator __first, _BidirectionalIterator __last, bidirectional_iterator_tag)`
- `template<typename _RandomAccessIterator >`  
`void std::__reverse (_RandomAccessIterator __first, _RandomAccessIterator __last, random_access_iterator_tag)`
- `template<typename _ForwardIterator >`  
`void std::__rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, forward_iterator_tag)`
- `template<typename _BidirectionalIterator >`  
`void std::__rotate (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, bidirectional_iterator_tag)`
- `template<typename _RandomAccessIterator >`  
`void std::__rotate (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, random_access_iterator_tag)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _Distance >`  
`_BidirectionalIterator1 std::__rotate_adaptive (_BidirectionalIterator1 __first, _BidirectionalIterator1 __middle, _BidirectionalIterator1 __last, _Distance __len1, _Distance __len2, _BidirectionalIterator2 __buffer, _Distance __buffer_size)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp >`  
`_ForwardIterator std::__search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val, std::forward_iterator_tag)`
- `template<typename _RandomAccessIter, typename _Integer, typename _Tp >`  
`_RandomAccessIter std::__search_n (_RandomAccessIter __first, _RandomAccessIter __last, _Integer __count, const _Tp &__val, std::random_access_iterator_tag)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`  
`_ForwardIterator std::__search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, std::forward_iterator_tag)`
- `template<typename _RandomAccessIter, typename _Integer, typename _Tp, typename _BinaryPredicate >`  
`_RandomAccessIter std::__search_n (_RandomAccessIter __first, _RandomAccessIter __last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, std::random_access_iterator_tag)`

- `template<typename _ForwardIterator, typename _Pointer, typename _Predicate, typename _Distance >`  
`_ForwardIterator std::\_\_stable\_partition\_adaptive (_ForwardIterator __first, _`  
`ForwardIterator __last, _Predicate __pred, _Distance __len, _Pointer __buffer,`  
`_Distance __buffer_size)`
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Distance >`  
`void std::__stable_sort_adaptive (_RandomAccessIterator __first, _`  
`RandomAccessIterator __last, _Pointer __buffer, _Distance __buffer_size)`
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Distance, typename`  
`_Compare >`  
`void std::__stable_sort_adaptive (_RandomAccessIterator __first, _`  
`RandomAccessIterator __last, _Pointer __buffer, _Distance __buffer_size,`  
`_Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::\_\_unguarded\_insertion\_sort (_RandomAccessIterator __first, _`  
`RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::\_\_unguarded\_insertion\_sort (_RandomAccessIterator __first, _`  
`RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::\_\_unguarded\_linear\_insert (_RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::\_\_unguarded\_linear\_insert (_RandomAccessIterator __last, _Compare`  
`__comp)`
- `template<typename _RandomAccessIterator, typename _Tp >`  
`_RandomAccessIterator std::\_\_unguarded\_partition (_RandomAccessIterator _`  
`__first, _RandomAccessIterator __last, const _Tp &__pivot)`
- `template<typename _RandomAccessIterator, typename _Tp, typename _Compare >`  
`_RandomAccessIterator std::\_\_unguarded\_partition (_RandomAccessIterator _`  
`__first, _RandomAccessIterator __last, const _Tp &__pivot, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`_RandomAccessIterator std::\_\_unguarded\_partition\_pivot (_`  
`RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`_RandomAccessIterator std::\_\_unguarded\_partition\_pivot (_`  
`RandomAccessIterator __first, _RandomAccessIterator __last, _Compare`  
`__comp)`
- `template<typename _ForwardIterator, typename _OutputIterator >`  
`_OutputIterator std::\_\_unique\_copy (_ForwardIterator __first, _ForwardIterator`  
`__last, _OutputIterator __result, forward_iterator_tag, output_iterator_tag)`
- `template<typename _InputIterator, typename _OutputIterator >`  
`_OutputIterator std::\_\_unique\_copy (_InputIterator __first, _InputIterator __last,`  
`_OutputIterator __result, input_iterator_tag, output_iterator_tag)`

- `template<typename _InputIterator, typename _ForwardIterator >`  
`_ForwardIterator std::\_\_unique\_copy (_InputIterator __first, _InputIterator __-`  
`last, _ForwardIterator __result, input_iterator_tag, forward_iterator_tag)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _BinaryPredicate >`  
`_OutputIterator std::\_\_unique\_copy (_ForwardIterator __first, _ForwardIterator`  
`__last, _OutputIterator __result, _BinaryPredicate __binary_pred, forward-`  
`iterator_tag, output_iterator_tag)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate >`  
`_OutputIterator std::\_\_unique\_copy (_InputIterator __first, _InputIterator __last,`  
`_OutputIterator __result, _BinaryPredicate __binary_pred, input_iterator_tag,`  
`output_iterator_tag)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`  
`_ForwardIterator std::\_\_unique\_copy (_InputIterator __first, _InputIterator __-`  
`last, _ForwardIterator __result, _BinaryPredicate __binary_pred, input_-`  
`iterator_tag, forward_iterator_tag)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator std::adjacent\_find (_ForwardIterator __first, _ForwardIterator`  
`__last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`  
`_ForwardIterator std::adjacent\_find (_ForwardIterator __first, _ForwardIterator`  
`__last, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool std::all\_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool std::any\_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Tp >`  
`bool std::binary\_search (_ForwardIterator __first, _ForwardIterator __last, const`  
`_Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`bool std::binary\_search (_ForwardIterator __first, _ForwardIterator __last, const`  
`_Tp &__val, _Compare __comp)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::copy\_if (_InputIterator __first, _InputIterator __last, _-`  
`OutputIterator __result, _Predicate __pred)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`  
`_OutputIterator std::copy\_n (_InputIterator __first, _Size __n, _OutputIterator`  
`__result)`
- `template<typename _InputIterator, typename _Tp >`  
`iterator_traits< _InputIterator >::difference_type std::count (_InputIterator __-`  
`first, _InputIterator __last, const _Tp &__value)`
- `template<typename _InputIterator, typename _Predicate >`  
`iterator_traits< _InputIterator >::difference_type std::count\_if (_InputIterator`  
`__first, _InputIterator __last, _Predicate __pred)`

- `template<typename _ForwardIterator, typename _Tp >`  
`pair< _ForwardIterator, _ForwardIterator > std::equal\_range (_ForwardIterator`  
`__first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`pair< _ForwardIterator, _ForwardIterator > std::equal\_range (_ForwardIterator`  
`__first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _InputIterator, typename _Tp >`  
`_InputIterator std::find (_InputIterator __first, _InputIterator __last, const _Tp`  
`&__val)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`_ForwardIterator1 std::find\_end (_ForwardIterator1 __first1, _ForwardIterator1`  
`__last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate`  
`>`  
`_ForwardIterator1 std::find\_end (_ForwardIterator1 __first1, _ForwardIterator1`  
`__last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _-`  
`BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _ForwardIterator >`  
`_InputIterator std::find\_first\_of (_InputIterator __first1, _InputIterator __last1,`  
`_ForwardIterator __first2, _ForwardIterator __last2)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`  
`_InputIterator std::find\_first\_of (_InputIterator __first1, _InputIterator __last1,`  
`_ForwardIterator __first2, _ForwardIterator __last2, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _Predicate >`  
`_InputIterator std::find\_if (_InputIterator __first, _InputIterator __last, _-`  
`Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate >`  
`_InputIterator std::find\_if\_not (_InputIterator __first, _InputIterator __last, _-`  
`Predicate __pred)`
- `template<typename _InputIterator, typename _Function >`  
`_Function std::for\_each (_InputIterator __first, _InputIterator __last, _Function`  
`__f)`
- `template<typename _ForwardIterator, typename _Generator >`  
`void std::generate (_ForwardIterator __first, _ForwardIterator __last, _Generator`  
`__gen)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`_OutputIterator std::generate\_n (_OutputIterator __first, _Size __n, _Generator`  
`__gen)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _-`  
`InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare >`  
`bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _-`  
`InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)`

- `template<typename _BidirectionalIterator >`  
`void std::inplace\_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`void std::inplace\_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool std::is\_partitioned (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`bool std::is\_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`bool std::is\_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _BinaryPredicate __pred)`
- `template<typename _ForwardIterator >`  
`bool std::is\_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`bool std::is\_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator std::is\_sorted\_until (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`_ForwardIterator std::is\_sorted\_until (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`_ForwardIterator std::lower\_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _Tp >`  
`_Tp std::max (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`  
`_Tp std::max (initializer_list< _Tp >, _Compare)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator std::max\_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`_ForwardIterator std::max\_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`

- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1,`  
`_InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _`  
`Compare __comp)`
- `template<typename _Tp, typename _Compare >`  
`_Tp std::min (initializer_list< _Tp >, _Compare)`
- `template<typename _Tp >`  
`_Tp std::min (initializer_list< _Tp >)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator std::min\_element (_ForwardIterator __first, _ForwardIterator`  
`__last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`_ForwardIterator std::min\_element (_ForwardIterator __first, _ForwardIterator`  
`__last, _Compare __comp)`
- `template<typename _Tp >`  
`pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp`  
`&__b)`
- `template<typename _Tp >`  
`pair< _Tp, _Tp > std::minmax (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`  
`pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp`  
`&__b, _Compare __comp)`
- `template<typename _Tp, typename _Compare >`  
`pair< _Tp, _Tp > std::minmax (initializer_list< _Tp >, _Compare)`
- `template<typename _ForwardIterator, typename _Compare >`  
`pair< _ForwardIterator, _ForwardIterator > std::minmax\_element (_`  
`ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator >`  
`pair< _ForwardIterator, _ForwardIterator > std::minmax\_element (_`  
`ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _BidirectionalIterator >`  
`bool std::next\_permutation (_BidirectionalIterator __first, _BidirectionalIterator`  
`__last)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`bool std::next\_permutation (_BidirectionalIterator __first, _BidirectionalIterator`  
`__last, _Compare __comp)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool std::none\_of (_InputIterator __first, _InputIterator __last, _Predicate __`  
`pred)`
- `template<typename _RandomAccessIterator >`  
`void std::nth\_element (_RandomAccessIterator __first, _RandomAccessIterator`  
`__nth, _RandomAccessIterator __last)`

- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator`  
`__nth, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator`  
`__middle, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator`  
`__middle, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _RandomAccessIterator >`  
`_RandomAccessIterator std::partial_sort_copy (_InputIterator __`  
`first, _InputIterator __last, _RandomAccessIterator __result_first, _`  
`RandomAccessIterator __result_last)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare >`  
`_RandomAccessIterator std::partial_sort_copy (_InputIterator __`  
`first, _InputIterator __last, _RandomAccessIterator __result_first, _`  
`RandomAccessIterator __result_last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator std::partition (_ForwardIterator __first, _ForwardIterator __`  
`last, _Predicate __pred)`
- `template<typename _InputIterator, typename _OutputIterator1, typename _OutputIterator2, type-`  
`name _Predicate >`  
`pair< _OutputIterator1, _OutputIterator2 > std::partition_copy (_InputIterator`  
`__first, _InputIterator __last, _OutputIterator1 __out_true, _OutputIterator2 __`  
`out_false, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator std::partition_point (_ForwardIterator __first, _ForwardIterator`  
`__last, _Predicate __pred)`
- `template<typename _BidirectionalIterator >`  
`bool std::prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator`  
`__last)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`bool std::prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator`  
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _RandomNumberGenerator >`  
`void std::random_shuffle (_RandomAccessIterator __first, _`  
`RandomAccessIterator __last, _RandomNumberGenerator &&__rand)`
- `template<typename _RandomAccessIterator >`  
`void std::random_shuffle (_RandomAccessIterator __first, _`  
`RandomAccessIterator __last)`
- `template<typename _ForwardIterator, typename _Tp >`  
`_ForwardIterator std::remove (_ForwardIterator __first, _ForwardIterator __last,`  
`const _Tp &__value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`  
`_OutputIterator std::remove_copy (_InputIterator __first, _InputIterator __last,`  
`_OutputIterator __result, const _Tp &__value)`

- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::remove\_copy\_if (_InputIterator __first, _InputIterator __-`  
`last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator std::remove\_if (_ForwardIterator __first, _ForwardIterator __-`  
`last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void std::replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp`  
`&__old_value, const _Tp &__new_value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`  
`_OutputIterator std::replace\_copy (_InputIterator __first, _InputIterator __last,`  
`_OutputIterator __result, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _`  
`Tp >`  
`_OutputIterator std::replace\_copy\_if (_InputIterator __first, _InputIterator __-`  
`last, _OutputIterator __result, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _ForwardIterator, typename _Predicate, typename _Tp >`  
`void std::replace\_if (_ForwardIterator __first, _ForwardIterator __last, _`  
`Predicate __pred, const _Tp &__new_value)`
- `template<typename _BidirectionalIterator >`  
`void std::reverse (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _OutputIterator >`  
`_OutputIterator std::reverse\_copy (_BidirectionalIterator __first, _`  
`BidirectionalIterator __last, _OutputIterator __result)`
- `template<typename _ForwardIterator >`  
`void std::rotate (_ForwardIterator __first, _ForwardIterator __middle, _`  
`ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _OutputIterator >`  
`_OutputIterator std::rotate\_copy (_ForwardIterator __first, _ForwardIterator __-`  
`middle, _ForwardIterator __last, _OutputIterator __result)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`_ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __-`  
`last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate`  
`>`  
`_ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __-`  
`last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate`  
`__predicate)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp >`  
`_ForwardIterator std::search\_n (_ForwardIterator __first, _ForwardIterator __-`  
`last, _Integer __count, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _`  
`BinaryPredicate >`  
`_ForwardIterator std::search\_n (_ForwardIterator __first, _ForwardIterator __-`  
`last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)`



- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator std::set\_difference (_InputIterator1 __first1, _InputIterator1 __-`  
`last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`  
`name _Compare >`  
`_OutputIterator std::set\_difference (_InputIterator1 __first1, _InputIterator1 __-`  
`last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result,`  
`_Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`  
`name _Compare >`  
`_OutputIterator std::set\_intersection (_InputIterator1 __first1, _InputIterator1`  
`__last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __-`  
`result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator std::set\_intersection (_InputIterator1 __first1, _InputIterator1`  
`__last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __-`  
`result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`  
`name _Compare >`  
`_OutputIterator std::set\_symmetric\_difference (_InputIterator1 __first1, _-`  
`InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _-`  
`OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator std::set\_symmetric\_difference (_InputIterator1 __first1, _-`  
`InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _-`  
`OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator std::set\_union (_InputIterator1 __first1, _InputIterator1 __last1,`  
`_InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`  
`name _Compare >`  
`_OutputIterator std::set\_union (_InputIterator1 __first1, _InputIterator1 __last1,`  
`_InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _-`  
`Compare __comp)`
- `template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator >`  
`void std::shuffle (_RandomAccessIterator __first, _RandomAccessIterator __-`  
`last, _UniformRandomNumberGenerator && __g)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last,`  
`_Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator std::stable\_partition (_ForwardIterator __first, _-`  
`ForwardIterator __last, _Predicate __pred)`

- `template<typename _RandomAccessIterator >`  
`void std::stable\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::stable\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::transform (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation >`  
`_OutputIterator std::transform (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`  
`_ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _InputIterator, typename _OutputIterator >`  
`_OutputIterator std::unique\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate >`  
`_OutputIterator std::unique\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred)`
- `template<typename _ForwardIterator, typename _Tp >`  
`_ForwardIterator std::upper\_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`_ForwardIterator std::upper\_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`

### 6.295.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<algorithm>`.

Definition in file [stl\\_algo.h](#).

## 6.296 `stl_algobase.h` File Reference

### Namespaces

- namespace [std](#)

**Defines**

- `#define _GLIBCXX_MOVE3(_Tp, _Up, _Vp)`
- `#define _GLIBCXX_MOVE_BACKWARD3(_Tp, _Up, _Vp)`

**Functions**

- `template<bool _IsMove, typename _II, typename _OI >`  
`_OI std::__copy_move_a (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_`  
`iterator< _CharT, char_traits< _CharT > > >::__type std::__copy_move_a2`  
`( _CharT *, _CharT *, ostreambuf_iterator< _CharT, char_traits< _CharT > > )`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_`  
`iterator< _CharT, char_traits< _CharT > > >::__type std::__copy_move_a2`  
`(const _CharT *, const _CharT *, ostreambuf_iterator< _CharT, char_traits<`  
`_CharT > > )`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type`  
`std::__copy_move_a2 (istreambuf_iterator< _CharT, char_traits< _CharT >`  
`>, istreambuf_iterator< _CharT, char_traits< _CharT > >, _CharT *)`
- `template<bool _IsMove, typename _II, typename _OI >`  
`_OI std::__copy_move_a2 (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _BI1, typename _BI2 >`  
`_BI2 std::__copy_move_backward_a (_BI1 __first, _BI1 __last, _BI2 __-`  
`result)`
- `template<bool _IsMove, typename _BI1, typename _BI2 >`  
`_BI2 std::__copy_move_backward_a2 (_BI1 __first, _BI1 __last, _BI2 __-`  
`result)`
- `template<typename _II1, typename _II2 >`  
`bool std::__equal_aux (_II1 __first1, _II1 __last1, _II2 __first2)`
- `template<typename _ForwardIterator, typename _Tp >`  
`__gnu_cxx::__enable_if< !__is_scalar< _Tp >::__value, void >::__type std::_-`  
`fill_a (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_byte< _Tp >::__value, void >::__type std::_-`  
`fill_a (_Tp *__first, _Tp *__last, const _Tp &__c)`
- `template<typename _OutputIterator, typename _Size, typename _Tp >`  
`__gnu_cxx::__enable_if< !__is_scalar< _Tp >::__value, _OutputIterator >::__-`  
`type std::__fill_n_a (_OutputIterator __first, _Size __n, const _Tp &__value)`

- `template<typename _Size, typename _Tp >`  
`__gnu_cxx::__enable_if< __is_byte< _Tp >::__value, _Tp * >::__type std::-`  
`fill_n_a` (`_Tp *``__first`, `_Size` `__n`, `const _Tp &``__c`)
- `template<typename _II1, typename _II2 >`  
`bool std::lexicographical_compare_aux` (`_II1` `__first1`, `_II1` `__last1`, `_II2` `__-`  
`first2`, `_II2` `__last2`)
- `long long std::__lg` (`long long` `__n`)
- `long std::__lg` (`long` `__n`)
- `template<typename _Size >`  
`_Size std::__lg` (`_Size` `__n`)
- `int std::__lg` (`int` `__n`)
- `template<typename _Iterator >`  
`_Miter_base< _Iterator >::iterator_type std::__miter_base` (`_Iterator` `__it`)
- `template<typename _Iterator >`  
`_Niter_base< _Iterator >::iterator_type std::__niter_base` (`_Iterator` `__it`)
- `template<typename _II, typename _OI >`  
`_OI std::copy` (`_II` `__first`, `_II` `__last`, `_OI` `__result`)
- `template<typename _BI1, typename _BI2 >`  
`_BI2 std::copy_backward` (`_BI1` `__first`, `_BI1` `__last`, `_BI2` `__result`)
- `template<typename _II1, typename _II2 >`  
`bool std::equal` (`_II1` `__first1`, `_II1` `__last1`, `_II2` `__first2`)
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`  
`bool std::equal` (`_Iter1` `__first1`, `_Iter1` `__last1`, `_Iter2` `__first2`, `-`  
`BinaryPredicate` `__binary_pred`)
- `template<typename _ForwardIterator, typename _Tp >`  
`void std::fill` (`_ForwardIterator` `__first`, `_ForwardIterator` `__last`, `const _Tp &``__-`  
`value`)
- `template<typename _OI, typename _Size, typename _Tp >`  
`_OI std::fill_n` (`_OI` `__first`, `_Size` `__n`, `const _Tp &``__value`)
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`void std::iter_swap` (`_ForwardIterator1` `__a`, `_ForwardIterator2` `__b`)
- `template<typename _II1, typename _II2, typename _Compare >`  
`bool std::lexicographical_compare` (`_II1` `__first1`, `_II1` `__last1`, `_II2` `__first2`, `_II2`  
`__last2`, `_Compare` `__comp`)
- `template<typename _II1, typename _II2 >`  
`bool std::lexicographical_compare` (`_II1` `__first1`, `_II1` `__last1`, `_II2` `__first2`, `_II2`  
`__last2`)
- `template<typename _ForwardIterator, typename _Tp >`  
`_ForwardIterator std::lower_bound` (`_ForwardIterator` `__first`, `_ForwardIterator`  
`__last`, `const _Tp &``__val`)
- `template<typename _Tp, typename _Compare >`  
`const _Tp & std::max` (`const _Tp &``__a`, `const _Tp &``__b`, `_Compare` `__comp`)
- `template<typename _Tp >`  
`const _Tp & std::max` (`const _Tp &``__a`, `const _Tp &``__b`)

- `template<typename _Tp, typename _Compare >`  
`const _Tp & std::min (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`  
`const _Tp & std::min (const _Tp &__a, const _Tp &__b)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`  
`pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2)`
- `template<typename _II, typename _OI >`  
`_OI std::move (_II __first, _II __last, _OI __result)`
- `template<typename _BI1, typename _BI2 >`  
`_BI2 std::move\_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`_ForwardIterator2 std::swap\_ranges (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`

### 6.296.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<algorithm>`.

Definition in file `std::algbase.h`.

## 6.297 `std::bvector.h` File Reference

### Classes

- struct [std::hash<::vector< bool, \\_Alloc > >](#)  
*[std::hash](#) specialization for `vector<bool>`.*
- class [std::vector< bool, \\_Alloc >](#)  
*A specialization of `vector` for booleans which offers fixed time access to individual elements in any order.*

### Namespaces

- namespace [std](#)

## Typedefs

- typedef unsigned long `std::_Bit_type`

## Enumerations

- enum { `_S_word_bit` }

## Functions

- void `std::_fill_bvector` (`_Bit_iterator` \_\_first, `_Bit_iterator` \_\_last, bool \_\_x)
- void `std::fill` (`_Bit_iterator` \_\_first, `_Bit_iterator` \_\_last, const bool &\_\_x)
- `_Bit_iterator` `std::operator+` (`ptrdiff_t` \_\_n, const `_Bit_iterator` &\_\_x)
- `_Bit_const_iterator` `std::operator+` (`ptrdiff_t` \_\_n, const `_Bit_const_iterator` &\_\_x)
- `ptrdiff_t` `std::operator-` (const `_Bit_iterator_base` &\_\_x, const `_Bit_iterator_base` &\_\_y)

### 6.297.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<vector>`.

Definition in file [std\\_bvector.h](#).

## 6.298 `std::construct.h` File Reference

### Namespaces

- namespace [std](#)

### Functions

- template<typename `_T1`, typename... `_Args`>  
void [std::\\_Construct](#) (`_T1` \*\_\_p, `_Args` &&...\_\_args)
- template<typename `_ForwardIterator`, typename `_Tp`>  
void `std::_Destroy` (`_ForwardIterator` \_\_first, `_ForwardIterator` \_\_last, `allocator<_Tp>` &)
- template<typename `_ForwardIterator`, typename `_Allocator`>  
void `std::_Destroy` (`_ForwardIterator` \_\_first, `_ForwardIterator` \_\_last, `_Allocator` &\_\_alloc)
- template<typename `_ForwardIterator`>  
void [std::\\_Destroy](#) (`_ForwardIterator` \_\_first, `_ForwardIterator` \_\_last)

- `template<typename _Tp >`  
`void std::_Destroy (_Tp * __pointer)`

### 6.298.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

Definition in file `std_construct.h`.

## 6.299 `std::deque.h` File Reference

### Classes

- class `std::_Deque_base< _Tp, _Alloc >`
- struct `std::_Deque_iterator< _Tp, _Ref, _Ptr >`  
*A deque::iterator.*
- class `std::deque< _Tp, _Alloc >`  
*A standard container using fixed-size memory allocation and constant-time manipulation of elements at either end.*

### Namespaces

- namespace `std`

### Defines

- `#define _GLIBCXX_DEQUE_BUF_SIZE`

### Functions

- `size_t std::__deque_buf_size (size_t __size)`
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::copy (_Deque_iterator< _Tp,`  
`const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const _Tp &, const`  
`_Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::copy (_Deque_iterator< _Tp, _Tp`  
`&, _Tp * > __first, _Deque_iterator< _Tp, _Tp &, _Tp * > __last, _Deque_-`  
`iterator< _Tp, _Tp &, _Tp * > __result)`

- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::copy_backward` (`_Deque_iterator< _Tp, const _Tp &, const _Tp * > __first`, `_Deque_iterator< _Tp, const _Tp &, const _Tp * > __last`, `_Deque_iterator< _Tp, _Tp &, _Tp * > __result`)
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::copy_backward` (`_Deque_iterator< _Tp, _Tp &, _Tp * > __first`, `_Deque_iterator< _Tp, _Tp &, _Tp * > __last`, `_Deque_iterator< _Tp, _Tp &, _Tp * > __result`)
- `template<typename _Tp >`  
`void std::fill` (`const _Deque_iterator< _Tp, _Tp &, _Tp * > &__first`, `const _Deque_iterator< _Tp, _Tp &, _Tp * > &__last`, `const _Tp &__value`)
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::move` (`_Deque_iterator< _Tp, const _Tp &, const _Tp * > __first`, `_Deque_iterator< _Tp, const _Tp &, const _Tp * > __last`, `_Deque_iterator< _Tp, _Tp &, _Tp * > __result`)
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::move` (`_Deque_iterator< _Tp, _Tp &, _Tp * > __first`, `_Deque_iterator< _Tp, _Tp &, _Tp * > __last`, `_Deque_iterator< _Tp, _Tp &, _Tp * > __result`)
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::move_backward` (`_Deque_iterator< _Tp, const _Tp &, const _Tp * > __first`, `_Deque_iterator< _Tp, const _Tp &, const _Tp * > __last`, `_Deque_iterator< _Tp, _Tp &, _Tp * > __result`)
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::move_backward` (`_Deque_iterator< _Tp, _Tp &, _Tp * > __first`, `_Deque_iterator< _Tp, _Tp &, _Tp * > __last`, `_Deque_iterator< _Tp, _Tp &, _Tp * > __result`)
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`bool std::operator!=` (`const _Deque_iterator< _Tp, _Ref, _Ptr > &__x`, `const _Deque_iterator< _Tp, _Ref, _Ptr > &__y`)
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`  
`bool std::operator!=` (`const _Deque_iterator< _Tp, _RefL, _PtrL > &__x`, `const _Deque_iterator< _Tp, _RefR, _PtrR > &__y`)
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator!=` (`const deque< _Tp, _Alloc > &__x`, `const deque< _Tp, _Alloc > &__y`)
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`_Deque_iterator< _Tp, _Ref, _Ptr > std::operator+` (`ptrdiff_t __n`, `const _Deque_iterator< _Tp, _Ref, _Ptr > &__x`)
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`_Deque_iterator< _Tp, _Ref, _Ptr >::difference_type std::operator-` (`const _Deque_iterator< _Tp, _Ref, _Ptr > &__x`, `const _Deque_iterator< _Tp, _Ref, _Ptr > &__y`)



- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR>`  
`>`  
`_Deque_iterator< _Tp, _RefL, _PtrL >::difference_type std::operator- (const`  
`_Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _`  
`RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR>`  
`>`  
`bool std::operator< (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x,`  
`const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator< (const deque< _Tp, _Alloc > &__x, const deque< _Tp,`  
`_Alloc > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`bool std::operator< (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const`  
`_Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR>`  
`>`  
`bool std::operator<= (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x,`  
`const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`bool std::operator<= (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const`  
`_Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator<= (const deque< _Tp, _Alloc > &__x, const deque< _Tp,`  
`_Alloc > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`bool std::operator== (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const`  
`_Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator== (const deque< _Tp, _Alloc > &__x, const deque< _Tp,`  
`_Alloc > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR>`  
`>`  
`bool std::operator== (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x,`  
`const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`bool std::operator> (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const`  
`_Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator> (const deque< _Tp, _Alloc > &__x, const deque< _Tp,`  
`_Alloc > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR>`  
`>`  
`bool std::operator> (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x,`  
`const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)`

- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`bool std::operator>= (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const`  
`_Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR`  
`>`  
`bool std::operator>= (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x,`  
`const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator>= (const deque< _Tp, _Alloc > &__x, const deque< _Tp,`  
`_Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`void std::swap (deque< _Tp, _Alloc > &__x, deque< _Tp, _Alloc > &__y)`

### 6.299.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<deque>`.

Definition in file [stl\\_deque.h](#).

### 6.299.2 Define Documentation

#### 6.299.2.1 `#define _GLIBCXX_DEQUE_BUF_SIZE`

This function controls the size of memory nodes.

#### Parameters

*size* The size of an element.

#### Returns

The number (not byte size) of elements per node.

This function started off as a compiler kludge from SGI, but seems to be a useful wrapper around a repeated constant expression. The **512** is tunable (and no other code needs to change), but no investigation has been done since inheriting the SGI code. Touch `_GLIBCXX_DEQUE_BUF_SIZE` only if you know what you are doing, however: changing it breaks the binary compatibility!!

Definition at line 84 of file `stl_deque.h`.

## 6.300 `std_function.h` File Reference

### Classes

- struct `std::binary_function<_Arg1, _Arg2, _Result>`
  - class `std::binary_negate<_Predicate>`  
*One of the [negation functors](#).*
- class `std::const_mem_fun1_ref_t<_Ret, _Tp, _Arg>`  
*One of the [adaptors for member /// pointers](#).*
- class `std::const_mem_fun1_t<_Ret, _Tp, _Arg>`  
*One of the [adaptors for member /// pointers](#).*
- class `std::const_mem_fun_ref_t<_Ret, _Tp>`  
*One of the [adaptors for member /// pointers](#).*
- class `std::const_mem_fun_t<_Ret, _Tp>`  
*One of the [adaptors for member /// pointers](#).*
- struct `std::divides<_Tp>`  
*One of the [math functors](#).*
- struct `std::equal_to<_Tp>`  
*One of the [comparison functors](#).*
- struct `std::greater<_Tp>`  
*One of the [comparison functors](#).*
- struct `std::greater_equal<_Tp>`  
*One of the [comparison functors](#).*
- struct `std::less<_Tp>`  
*One of the [comparison functors](#).*
- struct `std::less_equal<_Tp>`  
*One of the [comparison functors](#).*
- struct `std::logical_and<_Tp>`  
*One of the [Boolean operations functors](#).*
- struct `std::logical_not<_Tp>`  
*One of the [Boolean operations functors](#).*

- struct `std::logical_or<_Tp>`  
*One of the Boolean operations functors.*
- class `std::mem_fun1_ref_t<_Ret, _Tp, _Arg>`  
*One of the adaptors for member /// pointers.*
- class `std::mem_fun1_t<_Ret, _Tp, _Arg>`  
*One of the adaptors for member /// pointers.*
- class `std::mem_fun_ref_t<_Ret, _Tp>`  
*One of the adaptors for member /// pointers.*
- class `std::mem_fun_t<_Ret, _Tp>`  
*One of the adaptors for member /// pointers.*
- struct `std::minus<_Tp>`  
*One of the math functors.*
- struct `std::modulus<_Tp>`  
*One of the math functors.*
- struct `std::multiplies<_Tp>`  
*One of the math functors.*
- struct `std::negate<_Tp>`  
*One of the math functors.*
- struct `std::not_equal_to<_Tp>`  
*One of the comparison functors.*
- struct `std::plus<_Tp>`  
*One of the math functors.*
- class `std::pointer_to_binary_function<_Arg1, _Arg2, _Result>`  
*One of the adaptors for function pointers.*
- class `std::pointer_to_unary_function<_Arg, _Result>`  
*One of the adaptors for function pointers.*
- struct `std::unary_function<_Arg, _Result>`
- class `std::unary_negate<_Predicate>`  
*One of the negation functors.*

## Namespaces

- namespace [std](#)

## Functions

- `template<typename _Ret, typename _Tp >`  
`mem_fun_t< _Ret, _Tp > std::mem_fun (_Ret(_Tp::*__f)())`
- `template<typename _Ret, typename _Tp, typename _Arg >`  
`mem_fun1_t< _Ret, _Tp, _Arg > std::mem_fun (_Ret(_Tp::*__f)(_Arg))`
- `template<typename _Ret, typename _Tp, typename _Arg >`  
`mem_fun1_ref_t< _Ret, _Tp, _Arg > std::mem_fun_ref (_Ret(_Tp::*__f)(_-Arg))`
- `template<typename _Ret, typename _Tp >`  
`mem_fun_ref_t< _Ret, _Tp > std::mem_fun_ref (_Ret(_Tp::*__f)())`
- `template<typename _Predicate >`  
`unary_negate< _Predicate > std::not1 (const _Predicate &__pred)`
- `template<typename _Predicate >`  
`binary_negate< _Predicate > std::not2 (const _Predicate &__pred)`
- `template<typename _Arg, typename _Result >`  
`pointer_to_unary_function< _Arg, _Result > std::ptr\_fun (_Result(*__x)(_-Arg))`
- `template<typename _Arg1, typename _Arg2, typename _Result >`  
`pointer_to_binary_function< _Arg1, _Arg2, _Result > std::ptr\_fun (_Result(*__x)(_Arg1, _Arg2))`

### 6.300.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

Definition in file [std\\_function.h](#).

## 6.301 `std_heap.h` File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp >`  
`void std::adjust_heap (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __len, _Tp __value)`

- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _Compare >`  
`void std::__adjust_heap (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __len, _Tp __value, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance >`  
`bool std::__is_heap (_RandomAccessIterator __first, _Distance __n)`
- `template<typename _RandomAccessIterator, typename _Compare, typename _Distance >`  
`bool std::__is_heap (_RandomAccessIterator __first, _Compare __comp, _Distance __n)`
- `template<typename _RandomAccessIterator >`  
`bool std::__is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`bool std::__is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Compare >`  
`_Distance std::__is_heap_until (_RandomAccessIterator __first, _Distance __n, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance >`  
`_Distance std::__is_heap_until (_RandomAccessIterator __first, _Distance __n)`
- `template<typename _RandomAccessIterator >`  
`void std::__pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomAccessIterator __result)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::__pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomAccessIterator __result, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _Compare >`  
`void std::__push_heap (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __topIndex, _Tp __value, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp >`  
`void std::__push_heap (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __topIndex, _Tp __value)`
- `template<typename _RandomAccessIterator >`  
`bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`_RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last)`

- `template<typename _RandomAccessIterator, typename _Compare >`  
`_RandomAccessIterator std::is\_heap\_until (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::make\_heap (_RandomAccessIterator __first, _RandomAccessIterator`  
`__last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::make\_heap (_RandomAccessIterator __first, _RandomAccessIterator`  
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::pop\_heap (_RandomAccessIterator __first, _RandomAccessIterator _-`  
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::pop\_heap (_RandomAccessIterator __first, _RandomAccessIterator _-`  
`__last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::push\_heap (_RandomAccessIterator __first, _RandomAccessIterator`  
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::push\_heap (_RandomAccessIterator __first, _RandomAccessIterator`  
`__last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::sort\_heap (_RandomAccessIterator __first, _RandomAccessIterator _-`  
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`void std::sort\_heap (_RandomAccessIterator __first, _RandomAccessIterator _-`  
`__last)`

### 6.301.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<queue>`.

Definition in file [stl\\_heap.h](#).

## 6.302 `stl_iterator.h` File Reference

### Classes

- class [std::back\\_insert\\_iterator](#)< [\\_Container](#) >  
*Turns assignment into insertion.*
- class [std::front\\_insert\\_iterator](#)< [\\_Container](#) >  
*Turns assignment into insertion.*

- class `std::insert_iterator<_Container>`  
*Turns assignment into insertion.*
- class `std::move_iterator<_Iterator>`
- class `std::reverse_iterator<_Iterator>`

## Namespaces

- namespace `__gnu_cxx`
- namespace `std`

## Defines

- `#define _GLIBCXX_MAKE_MOVE_ITERATOR(_Iter)`

## Functions

- `template<typename _Container>`  
`back_insert_iterator<_Container> std::back_inserter (_Container &__x)`
- `template<typename _Container>`  
`front_insert_iterator<_Container> std::front_inserter (_Container &__x)`
- `template<typename _Container, typename _Iterator>`  
`insert_iterator<_Container> std::inserter (_Container &__x, _Iterator __i)`
- `template<typename _Iterator>`  
`move_iterator<_Iterator> std::make_move_iterator (const _Iterator &__i)`
- `template<typename _IteratorL, typename _IteratorR>`  
`bool std::operator!= (const reverse_iterator<_IteratorL> &__x, const reverse_iterator<_IteratorR> &__y)`
- `template<typename _Iterator>`  
`bool std::operator!= (const reverse_iterator<_Iterator> &__x, const reverse_iterator<_Iterator> &__y)`
- `template<typename _IteratorL, typename _IteratorR>`  
`bool std::operator!= (const move_iterator<_IteratorL> &__x, const move_iterator<_IteratorR> &__y)`
- `template<typename _Iterator>`  
`bool std::operator!= (const move_iterator<_Iterator> &__x, const move_iterator<_Iterator> &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container>`  
`bool __gnu_cxx::operator!= (const __normal_iterator<_IteratorL, _Container> &__lhs, const __normal_iterator<_IteratorR, _Container> &__rhs)`



- `template<typename _Iterator, typename _Container >`  
`bool __gnu_cxx::operator!= (const __normal_iterator< _Iterator, _Container`  
`> &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _Iterator, typename _Container >`  
`__normal_iterator< _Iterator, _Container > __gnu_cxx::operator+ (typename`  
`__normal_iterator< _Iterator, _Container >::difference_type __n, const __-`  
`normal_iterator< _Iterator, _Container > &__i)`
- `template<typename _Iterator >`  
`reverse_iterator< _Iterator > std::operator+ (typename reverse_iterator< _`  
`Iterator >::difference_type __n, const reverse_iterator< _Iterator > &__x)`
- `template<typename _Iterator >`  
`move_iterator< _Iterator > std::operator+ (typename move_iterator< _Iterator`  
`>::difference_type __n, const move_iterator< _Iterator > &__x)`
- `template<typename _Iterator, typename _Container >`  
`__normal_iterator< _Iterator, _Container >::difference_type __gnu -`  
`cxx::operator- (const __normal_iterator< _Iterator, _Container > &__lhs,`  
`const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR >`  
`auto std::operator- (const move_iterator< _IteratorL > &__x, const move_-`  
`iterator< _IteratorR > &__y)-> decltype(__x.base()-__y.base())`
- `template<typename _Iterator >`  
`auto std::operator- (const move_iterator< _Iterator > &__x, const move_-`  
`iterator< _Iterator > &__y)-> decltype(__x.base()-__y.base())`
- `template<typename _Iterator >`  
`reverse_iterator< _Iterator >::difference_type std::operator- (const reverse_-`  
`iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`auto std::operator- (const reverse_iterator< _IteratorL > &__x, const reverse_-`  
`iterator< _IteratorR > &__y)-> decltype(__y.base()-__x.base())`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`auto __gnu_cxx::operator- (const __normal_iterator< _IteratorL, _Container`  
`> &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)->`  
`decltype(__lhs.base()-__rhs.base())`
- `template<typename _Iterator >`  
`bool std::operator< (const move_iterator< _Iterator > &__x, const move_-`  
`iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool __gnu_cxx::operator< (const __normal_iterator< _IteratorL, _Container`  
`> &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator< (const move_iterator< _IteratorL > &__x, const move_-`  
`iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator< (const reverse_iterator< _IteratorL > &__x, const`  
`reverse_iterator< _IteratorR > &__y)`

- `template<typename _Iterator, typename _Container >`  
`bool __gnu_cxx::operator< (const __normal_iterator< _Iterator, _Container`  
`> &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _Iterator >`  
`bool std::operator< (const reverse_iterator< _Iterator > &__x, const reverse_`  
`iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator<= (const reverse_iterator< _IteratorL > &__x, const`  
`reverse_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`bool std::operator<= (const reverse_iterator< _Iterator > &__x, const`  
`reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator<= (const move_iterator< _IteratorL > &__x, const move_`  
`iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`bool std::operator<= (const move_iterator< _Iterator > &__x, const move_`  
`iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool __gnu_cxx::operator<= (const __normal_iterator< _IteratorL, _`  
`Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &_`  
`__rhs)`
- `template<typename _Iterator, typename _Container >`  
`bool __gnu_cxx::operator<= (const __normal_iterator< _Iterator, _Container`  
`> &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator== (const reverse_iterator< _IteratorL > &__x, const`  
`reverse_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`bool std::operator== (const reverse_iterator< _Iterator > &__x, const reverse_`  
`iterator< _Iterator > &__y)`
- `template<typename _Iterator, typename _Container >`  
`bool __gnu_cxx::operator== (const __normal_iterator< _Iterator, _Container`  
`> &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _Iterator >`  
`bool std::operator== (const move_iterator< _Iterator > &__x, const move_`  
`iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool __gnu_cxx::operator== (const __normal_iterator< _IteratorL, _`  
`Container > &__lhs, const __normal_iterator< _IteratorR, _Container >`  
`&__rhs)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator== (const move_iterator< _IteratorL > &__x, const move_`  
`iterator< _IteratorR > &__y)`

- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator> (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool gnu_cxx::operator> (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)`
- `template<typename _Iterator >`  
`bool std::operator> (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator> (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`bool std::operator> (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _Iterator, typename _Container >`  
`bool gnu_cxx::operator> (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _Iterator, typename _Container >`  
`bool gnu_cxx::operator>= (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator>= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool gnu_cxx::operator>= (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool std::operator>= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`bool std::operator>= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _Iterator >`  
`bool std::operator>= (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`

### 6.302.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

This file implements `reverse_iterator`, `back_insert_iterator`, `front_insert_iterator`, `insert_iterator`, `__normal_iterator`, and their supporting functions and overloaded operators.

Definition in file [std\\_iterator.h](#).

## 6.303 `std_iterator_base_funcs.h` File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _InputIterator, typename _Distance>`  
`void std::__advance (_InputIterator &__i, _Distance __n, input_iterator_tag)`
- `template<typename _BidirectionalIterator, typename _Distance>`  
`void std::__advance (_BidirectionalIterator &__i, _Distance __n, bidirectional_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Distance>`  
`void std::__advance (_RandomAccessIterator &__i, _Distance __n, random_access_iterator_tag)`
- `template<typename _RandomAccessIterator>`  
`iterator_traits< _RandomAccessIterator >::difference_type std::__distance (_RandomAccessIterator __first, _RandomAccessIterator __last, random_access_iterator_tag)`
- `template<typename _InputIterator>`  
`iterator_traits< _InputIterator >::difference_type std::__distance (_InputIterator __first, _InputIterator __last, input_iterator_tag)`
- `template<typename _InputIterator, typename _Distance>`  
`void std::advance (_InputIterator &__i, _Distance __n)`
- `template<typename _InputIterator>`  
`iterator_traits< _InputIterator >::difference_type std::distance (_InputIterator __first, _InputIterator __last)`
- `template<typename _ForwardIterator>`  
`_ForwardIterator std::next (_ForwardIterator __x, typename iterator_traits< _ForwardIterator >::difference_type __n=1)`
- `template<typename _BidirectionalIterator>`  
`_BidirectionalIterator std::prev (_BidirectionalIterator __x, typename iterator_traits< _BidirectionalIterator >::difference_type __n=1)`

### 6.303.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

This file contains all of the general iterator-related utility functions, such as [distance\(\)](#) and [advance\(\)](#).

Definition in file [std\\_iterator\\_base\\_funcs.h](#).

## 6.304 `std_iterator_base_types.h` File Reference

### Classes

- class [std::\\_\\_has\\_iterator\\_category\\_helper< \\_Tp >](#)  
*Traits class for iterators.*
- struct [std::bidirectional\\_iterator\\_tag](#)  
*Bidirectional iterators support a superset of forward iterator /// operations.*
- struct [std::forward\\_iterator\\_tag](#)  
*Forward iterators support a superset of input iterator operations.*
- struct [std::input\\_iterator\\_tag](#)  
*Marking input iterators.*
- struct [std::iterator< \\_Category, \\_Tp, \\_Distance, \\_Pointer, \\_Reference >](#)  
*Common iterator class.*
- struct [std::iterator\\_traits< \\_Tp \\* >](#)  
*Partial specialization for pointer types.*
- struct [std::iterator\\_traits< const \\_Tp \\* >](#)  
*Partial specialization for const pointer types.*
- struct [std::output\\_iterator\\_tag](#)  
*Marking output iterators.*
- struct [std::random\\_access\\_iterator\\_tag](#)  
*Random-access iterators support a superset of bidirectional /// iterator operations.*

## Namespaces

- namespace `std`

## Functions

- `template<typename _Iter >  
iterator_traits< _Iter >::iterator_category` `std::__iterator_category` (const `_Iter` &)

### 6.304.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

This file contains all of the general iterator-related utility types, such as `iterator_traits` and `struct iterator`.

Definition in file `std_iterator_base_types.h`.

## 6.305 `std::list.h` File Reference

### Classes

- struct `std::__detail::_List_node_base`  
*Common part of a node in the list.*
- class `std::_List_base< _Tp, _Alloc >`  
*See `bits/stl_deque.h`'s `_Deque_base` for an explanation.*
- struct `std::_List_const_iterator< _Tp >`  
*A `list::const_iterator`.*
- struct `std::_List_iterator< _Tp >`  
*A `list::iterator`.*
- struct `std::_List_node< _Tp >`  
*An actual node in the list.*
- class `std::list< _Tp, _Alloc >`  
*A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.*

## Namespaces

- namespace `std`
- namespace `std::__detail`

## Functions

- `template<typename _Val >`  
`bool std::operator!= (const _List_iterator< _Val > &__x, const _List_const_iterator< _Val > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator!= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator< (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator<= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Val >`  
`bool std::operator== (const _List_iterator< _Val > &__x, const _List_const_iterator< _Val > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator== (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator> (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator>= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`void std::swap (list< _Tp, _Alloc > &__x, list< _Tp, _Alloc > &__y)`

### 6.305.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include

Definition in file [std\\_list.h](#).

## 6.306 `std_map.h` File Reference

### Classes

- class `std::map<_Key, _Tp, _Compare, _Alloc>`  
*A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.*

### Namespaces

- namespace `std`

### Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`bool std::operator!= (const map<_Key, _Tp, _Compare, _Alloc> &__x, const map<_Key, _Tp, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`bool std::operator< (const map<_Key, _Tp, _Compare, _Alloc> &__x, const map<_Key, _Tp, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`bool std::operator<= (const map<_Key, _Tp, _Compare, _Alloc> &__x, const map<_Key, _Tp, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`bool std::operator== (const map<_Key, _Tp, _Compare, _Alloc> &__x, const map<_Key, _Tp, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`bool std::operator> (const map<_Key, _Tp, _Compare, _Alloc> &__x, const map<_Key, _Tp, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`bool std::operator>= (const map<_Key, _Tp, _Compare, _Alloc> &__x, const map<_Key, _Tp, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`void std::swap (map<_Key, _Tp, _Compare, _Alloc> &__x, map<_Key, _Tp, _Compare, _Alloc> &__y)`

#### 6.306.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<map>`.

Definition in file `std_map.h`.



## 6.307 `std_multimap.h` File Reference

### Classes

- class `std::multimap<_Key, _Tp, _Compare, _Alloc >`  
*A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.*

### Namespaces

- namespace `std`

### Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator!= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x,`  
`const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator< (const multimap< _Key, _Tp, _Compare, _Alloc > &__x,`  
`const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator<= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x,`  
`const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator== (const multimap< _Key, _Tp, _Compare, _Alloc > &__x,`  
`const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator> (const multimap< _Key, _Tp, _Compare, _Alloc > &__x,`  
`const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator>= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x,`  
`const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`void std::swap (multimap< _Key, _Tp, _Compare, _Alloc > &__x, multimap<`  
`_Key, _Tp, _Compare, _Alloc > &__y)`

#### 6.307.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<map>`.

Definition in file `std_multimap.h`.

## 6.308 `std_multiset.h` File Reference

### Classes

- class `std::multiset< _Key, _Compare, _Alloc >`  
*A standard container made up of elements, which can be retrieved in logarithmic time.*

### Namespaces

- namespace `std`

### Functions

- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator!= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator< (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator<= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator== (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator> (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator>= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`void std::swap (multiset< _Key, _Compare, _Alloc > &__x, multiset< _Key, _Compare, _Alloc > &__y)`

#### 6.308.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<set>`.

Definition in file `std_multiset.h`.

## 6.309 `std_numeric.h` File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _InputIterator, typename _Tp >`  
`_Tp std::accumulate (_InputIterator __first, _InputIterator __last, _Tp __init)`
- `template<typename _InputIterator, typename _Tp, typename _BinaryOperation >`  
`_Tp std::accumulate (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __binary_op)`
- `template<typename _InputIterator, typename _OutputIterator >`  
`_OutputIterator std::adjacent\_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::adjacent\_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _BinaryOperation1, typename _BinaryOperation2 >`  
`_Tp std::inner\_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init, _BinaryOperation1 __binary_op1, _BinaryOperation2 __binary_op2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp >`  
`_Tp std::inner\_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void std::iota (_ForwardIterator __first, _ForwardIterator __last, _Tp __value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::partial\_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _InputIterator, typename _OutputIterator >`  
`_OutputIterator std::partial\_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`

### 6.309.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<numeric>`.

Definition in file [std\\_numeric.h](#).

## 6.310 `std_pair.h` File Reference

### Classes

- struct `std::pair<_T1, _T2>`  
*Struct holding two objects of arbitrary type.*
- struct `std::piecewise_construct_t`  
*`piecewise_construct_t`*

### Namespaces

- namespace `std`

### Functions

- template<class `_T1`, class `_T2`>  
pair< typename `__decay_and_strip<_T1>::__type`, typename `__decay_and_strip<_T2>::__type`> `std::make_pair` (`_T1` &&`__x`, `_T2` &&`__y`)
- template<class `_T1`, class `_T2`>  
constexpr bool `std::operator!=` (const pair< `_T1`, `_T2`> &`__x`, const pair< `_T1`, `_T2`> &`__y`)
- template<class `_T1`, class `_T2`>  
constexpr bool `std::operator<` (const pair< `_T1`, `_T2`> &`__x`, const pair< `_T1`, `_T2`> &`__y`)
- template<class `_T1`, class `_T2`>  
constexpr bool `std::operator<=` (const pair< `_T1`, `_T2`> &`__x`, const pair< `_T1`, `_T2`> &`__y`)
- template<class `_T1`, class `_T2`>  
constexpr bool `std::operator==` (const pair< `_T1`, `_T2`> &`__x`, const pair< `_T1`, `_T2`> &`__y`)
- template<class `_T1`, class `_T2`>  
constexpr bool `std::operator>` (const pair< `_T1`, `_T2`> &`__x`, const pair< `_T1`, `_T2`> &`__y`)
- template<class `_T1`, class `_T2`>  
constexpr bool `std::operator>=` (const pair< `_T1`, `_T2`> &`__x`, const pair< `_T1`, `_T2`> &`__y`)
- template<class `_T1`, class `_T2`>  
void `std::swap` (pair< `_T1`, `_T2`> &`__x`, pair< `_T1`, `_T2`> &`__y`)

### Variables

- constexpr `piecewise_construct_t` `std::piecewise_construct`

### 6.310.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<utility>`.

Definition in file [std\\_pair.h](#).

## 6.311 `std_queue.h` File Reference

### Classes

- class [std::priority\\_queue<\\_Tp, \\_Sequence, \\_Compare>](#)  
*A standard container automatically sorting its contents.*
- class [std::queue<\\_Tp, \\_Sequence>](#)  
*A standard container giving FIFO behavior.*

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _Tp, typename _Seq>`  
`bool std::operator!= (const queue<_Tp, _Seq> &__x, const queue<_Tp, _Seq> &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool std::operator< (const queue<_Tp, _Seq> &__x, const queue<_Tp, _Seq> &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool std::operator<= (const queue<_Tp, _Seq> &__x, const queue<_Tp, _Seq> &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool std::operator== (const queue<_Tp, _Seq> &__x, const queue<_Tp, _Seq> &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool std::operator> (const queue<_Tp, _Seq> &__x, const queue<_Tp, _Seq> &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool std::operator>= (const queue<_Tp, _Seq> &__x, const queue<_Tp, _Seq> &__y)`

- `template<typename _Tp, typename _Sequence, typename _Compare >`  
`void std::swap (priority_queue< _Tp, _Sequence, _Compare > &__x,`  
`priority_queue< _Tp, _Sequence, _Compare > &__y)`
- `template<typename _Tp, typename _Seq >`  
`void std::swap (queue< _Tp, _Seq > &__x, queue< _Tp, _Seq > &__y)`

### 6.311.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<queue>`.

Definition in file [std\\_queue.h](#).

## 6.312 `std_raw_storage_iter.h` File Reference

### Classes

- class [std::raw\\_storage\\_iterator< \\_OutputIterator, \\_Tp >](#)

### Namespaces

- namespace [std](#)

### 6.312.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

Definition in file [std\\_raw\\_storage\\_iter.h](#).

## 6.313 `std_relops.h` File Reference

### Namespaces

- namespace [std](#)
- namespace [std::rel\\_ops](#)

### Functions

- `template<class _Tp >`  
`bool std::rel_ops::operator!= (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp >`  
`bool std::rel_ops::operator<= (const _Tp &__x, const _Tp &__y)`

- `template<class _Tp >`  
`bool std::rel_ops::operator> (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp >`  
`bool std::rel_ops::operator>= (const _Tp &__x, const _Tp &__y)`

### 6.313.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<utility>`.

Inclusion of this file has been removed from all of the other STL headers for safety reasons, except `std_utility.h`. For more information, see the thread of about twenty messages starting with <http://gcc.gnu.org/ml/libstdc++/2001-01/msg00223.html>, or [http://gcc.gnu.org/onlinedocs/libstdc++/faq.html#faq.ambiguous\\_overloads](http://gcc.gnu.org/onlinedocs/libstdc++/faq.html#faq.ambiguous_overloads)

Short summary: the `rel_ops` operators should be avoided for the present.

Definition in file `stl_relops.h`.

## 6.314 `stl_set.h` File Reference

### Classes

- class `std::set< _Key, _Compare, _Alloc >`  
*A standard container made up of unique keys, which can be retrieved in logarithmic time.*

### Namespaces

- namespace `std`

### Functions

- `template<typename _Key , typename _Compare , typename _Alloc >`  
`bool std::operator!= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key , typename _Compare , typename _Alloc >`  
`bool std::operator< (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key , typename _Compare , typename _Alloc >`  
`bool std::operator<= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`

- `template<typename _Key , typename _Compare , typename _Alloc >`  
`bool std::operator== (const set< _Key, _Compare, _Alloc > &__x, const set<`  
`_Key, _Compare, _Alloc > &__y)`
- `template<typename _Key , typename _Compare , typename _Alloc >`  
`bool std::operator> (const set< _Key, _Compare, _Alloc > &__x, const set<`  
`_Key, _Compare, _Alloc > &__y)`
- `template<typename _Key , typename _Compare , typename _Alloc >`  
`bool std::operator>= (const set< _Key, _Compare, _Alloc > &__x, const set<`  
`_Key, _Compare, _Alloc > &__y)`
- `template<typename _Key , typename _Compare , typename _Alloc >`  
`void std::swap (set< _Key, _Compare, _Alloc > &__x, set< _Key, _Compare,`  
`_Alloc > &__y)`

### 6.314.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<set>`.

Definition in file `std_set.h`.

## 6.315 `std_stack.h` File Reference

### Classes

- class `std::stack< _Tp, _Sequence >`  
*A standard container giving FILO behavior.*

### Namespaces

- namespace `std`

### Functions

- `template<typename _Tp , typename _Seq >`  
`bool std::operator!= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq`  
`> &__y)`
- `template<typename _Tp , typename _Seq >`  
`bool std::operator< (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq`  
`> &__y)`
- `template<typename _Tp , typename _Seq >`  
`bool std::operator<= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq`  
`> &__y)`



- `template<typename _Tp, typename _Seq >`  
`bool std::operator== (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool std::operator> (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool std::operator>= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`void std::swap (stack< _Tp, _Seq > &__x, stack< _Tp, _Seq > &__y)`

### 6.315.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<stack>`.

Definition in file [std::stack.h](#).

## 6.316 `std::tempbuf.h` File Reference

### Classes

- class [std::Temporary\\_buffer<\\_ForwardIterator, \\_Tp >](#)

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _ForwardIterator, typename _Tp >`  
`void std::__uninitialized_construct_buf (_ForwardIterator __first, _ForwardIterator __last, _Tp &__value)`
- `template<typename _Tp >`  
`pair< _Tp *, ptrdiff_t > std::get_temporary_buffer (ptrdiff_t __len)`
- `template<typename _Tp >`  
`void std::return_temporary_buffer (_Tp *__p)`

### 6.316.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

Definition in file [std\\_tempbuf.h](#).

## 6.317 `std_tree.h` File Reference

### Namespaces

- namespace [std](#)

### Enumerations

- enum `_Rb_tree_color` { `_S_red`, `_S_black` }

### Functions

- unsigned int **std::Rb\_tree\_black\_count** (const `_Rb_tree_node_base` \*\_\_node, const `_Rb_tree_node_base` \*\_\_root) throw ()
- `_Rb_tree_node_base` \* **std::Rb\_tree\_decrement** (`_Rb_tree_node_base` \*\_\_x) throw ()
- const `_Rb_tree_node_base` \* **std::Rb\_tree\_decrement** (const `_Rb_tree_node_base` \*\_\_x) throw ()
- const `_Rb_tree_node_base` \* **std::Rb\_tree\_increment** (const `_Rb_tree_node_base` \*\_\_x) throw ()
- `_Rb_tree_node_base` \* **std::Rb\_tree\_increment** (`_Rb_tree_node_base` \*\_\_x) throw ()
- void **std::Rb\_tree\_insert\_and\_rebalance** (const bool \_\_insert\_left, `_Rb_tree_node_base` \*\_\_x, `_Rb_tree_node_base` \*\_\_p, `_Rb_tree_node_base` &\_\_header) throw ()
- `_Rb_tree_node_base` \* **std::Rb\_tree\_rebalance\_for\_erase** (`_Rb_tree_node_base` \*const \_\_z, `_Rb_tree_node_base` &\_\_header) throw ()
- template<typename `_Key` , typename `_Val` , typename `_KeyOfValue` , typename `_Compare` , typename `_Alloc` >  
bool **std::operator!=** (const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > &\_\_x, const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > &\_\_y)
- template<typename `_Val` >  
bool **std::operator!=** (const `_Rb_tree_iterator`< `_Val` > &\_\_x, const `_Rb_tree_const_iterator`< `_Val` > &\_\_y)
- template<typename `_Key` , typename `_Val` , typename `_KeyOfValue` , typename `_Compare` , typename `_Alloc` >  
bool **std::operator<** (const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > &\_\_x, const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > &\_\_y)

- `template<typename _Key , typename _Val , typename _KeyOfValue , typename _Compare , typename _Alloc >`  
`bool std::operator<= (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Key , typename _Val , typename _KeyOfValue , typename _Compare , typename _Alloc >`  
`bool std::operator== (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Val >`  
`bool std::operator== (const _Rb_tree_iterator< _Val > &__x, const _Rb_tree_const_iterator< _Val > &__y)`
- `template<typename _Key , typename _Val , typename _KeyOfValue , typename _Compare , typename _Alloc >`  
`bool std::operator> (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Key , typename _Val , typename _KeyOfValue , typename _Compare , typename _Alloc >`  
`bool std::operator>= (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Key , typename _Val , typename _KeyOfValue , typename _Compare , typename _Alloc >`  
`void std::swap (_Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`

### 6.317.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<map>` or `<set>`.

Definition in file [std\\_tree.h](#).

## 6.318 `std_uninitialized.h` File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _InputIterator , typename _ForwardIterator , typename _Allocator >`

- ```

_FwdIterator std::__uninitialized_copy_a (_InputIterator __first, _
InputIterator __last, _ForwardIterator __result, _Allocator &__alloc)

```
- `template<typename _InputIterator, typename _ForwardIterator, typename _Tp>`
`_ForwardIterator std::__uninitialized_copy_a (_InputIterator __first, _`
`InputIterator __last, _ForwardIterator __result, allocator<_Tp> &)`
 - `template<typename _InputIterator1, typename _InputIterator2, typename _ForwardIterator, type-`
`name _Allocator >`
`_ForwardIterator std::__uninitialized_copy_move (_InputIterator1 __first1, _`
`InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _`
`ForwardIterator __result, _Allocator &__alloc)`
 - `template<typename _InputIterator, typename _Size, typename _ForwardIterator >`
`_ForwardIterator std::__uninitialized_copy_n (_InputIterator __first, _Size _`
`_n, _ForwardIterator __result, input_iterator_tag)`
 - `template<typename _RandomAccessIterator, typename _Size, typename _ForwardIterator >`
`_ForwardIterator std::__uninitialized_copy_n (_RandomAccessIterator __`
`first, _Size __n, _ForwardIterator __result, random_access_iterator_tag)`
 - `template<typename _ForwardIterator >`
`void std::__uninitialized_default (_ForwardIterator __first, _ForwardIterator`
`__last)`
 - `template<typename _ForwardIterator, typename _Allocator >`
`void std::__uninitialized_default_a (_ForwardIterator __first, _`
`ForwardIterator __last, _Allocator &__alloc)`
 - `template<typename _ForwardIterator, typename _Tp >`
`void std::__uninitialized_default_a (_ForwardIterator __first, _`
`ForwardIterator __last, allocator<_Tp> &)`
 - `template<typename _ForwardIterator, typename _Size >`
`void std::__uninitialized_default_n (_ForwardIterator __first, _Size __n)`
 - `template<typename _ForwardIterator, typename _Size, typename _Allocator >`
`void std::__uninitialized_default_n_a (_ForwardIterator __first, _Size __n, _`
`Allocator &__alloc)`
 - `template<typename _ForwardIterator, typename _Size, typename _Tp >`
`void std::__uninitialized_default_n_a (_ForwardIterator __first, _Size __n,`
`allocator<_Tp> &)`
 - `template<typename _ForwardIterator, typename _Tp, typename _Tp2 >`
`void std::__uninitialized_fill_a (_ForwardIterator __first, _ForwardIterator __`
`last, const _Tp &__x, allocator<_Tp2> &)`
 - `template<typename _ForwardIterator, typename _Tp, typename _Allocator >`
`void std::__uninitialized_fill_a (_ForwardIterator __first, _ForwardIterator __`
`last, const _Tp &__x, _Allocator &__alloc)`
 - `template<typename _ForwardIterator, typename _Tp, typename _InputIterator, typename _`
`Allocator >`
`_ForwardIterator std::__uninitialized_fill_move (_ForwardIterator __result, _`
`ForwardIterator __mid, const _Tp &__x, _InputIterator __first, _InputIterator`
`__last, _Allocator &__alloc)`

- `template<typename _ForwardIterator, typename _Size, typename _Tp, typename _Allocator >`
`void std::__uninitialized_fill_n_a (_ForwardIterator __first, _Size __n, const`
`_Tp &__x, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp, typename _Tp2 >`
`void std::__uninitialized_fill_n_a (_ForwardIterator __first, _Size __n, const`
`_Tp &__x, allocator< _Tp2 > &)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator >`
`_ForwardIterator std::__uninitialized_move_a (_InputIterator __first, _`
`InputIterator __last, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _ForwardIterator, type-`
`name _Allocator >`
`_ForwardIterator std::__uninitialized_move_copy (_InputIterator1 __first1, _`
`InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _`
`ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Tp, typename _`
`Allocator >`
`void std::__uninitialized_move_fill (_InputIterator __first1, _InputIterator __`
`last1, _ForwardIterator __first2, _ForwardIterator __last2, const _Tp &__x, _`
`Allocator &__alloc)`
- `template<typename _InputIterator, typename _ForwardIterator >`
`_ForwardIterator std::uninitialized_copy (_InputIterator __first, _InputIterator`
`__last, _ForwardIterator __result)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator >`
`_ForwardIterator std::uninitialized_copy_n (_InputIterator __first, _Size __n, _`
`ForwardIterator __result)`
- `template<typename _ForwardIterator, typename _Tp >`
`void std::uninitialized_fill (_ForwardIterator __first, _ForwardIterator __last,`
`const _Tp &__x)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp >`
`void std::uninitialized_fill_n (_ForwardIterator __first, _Size __n, const _Tp &_`
`__x)`

6.318.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

Definition in file [std_uninitialized.h](#).

6.319 `std_vector.h` File Reference

Classes

- struct [std::_Vector_base< _Tp, _Alloc >](#)

See [bits/stl_deque.h](#)'s `_Deque_base` for an explanation.

- class `std::vector<_Tp, _Alloc>`

A standard container which offers fixed time access to individual elements in any order.

Namespaces

- namespace `std`

Functions

- `template<typename _Tp, typename _Alloc>`
`bool std::operator!= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc>`
`bool std::operator< (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc>`
`bool std::operator<= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc>`
`bool std::operator== (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc>`
`bool std::operator> (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc>`
`bool std::operator>= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc>`
`void std::swap (vector< _Tp, _Alloc > &__x, vector< _Tp, _Alloc > &__y)`

6.319.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<vector>`.

Definition in file [std_vector.h](#).

6.320 `stream_iterator.h` File Reference

Classes

- class `std::istream_iterator< _Tp, _CharT, _Traits, _Dist >`
Provides input iterator semantics for streams.
- class `std::ostream_iterator< _Tp, _CharT, _Traits >`
Provides output iterator semantics for streams.

Namespaces

- namespace `std`

Functions

- `template<class _Tp, class _CharT, class _Traits, class _Dist >`
`bool std::operator!= (const istream_iterator< _Tp, _CharT, _Traits, _Dist > &_x, const istream_iterator< _Tp, _CharT, _Traits, _Dist > &_y)`
- `template<typename _Tp, typename _CharT, typename _Traits, typename _Dist >`
`bool std::operator== (const istream_iterator< _Tp, _CharT, _Traits, _Dist > &_x, const istream_iterator< _Tp, _CharT, _Traits, _Dist > &_y)`

6.320.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

Definition in file `stream_iterator.h`.

6.321 `streambuf` File Reference

Classes

- class `std::basic_streambuf< _CharT, _Traits >`
*The actual work of input and output (interface).
This is a base class. Derived stream buffers each control a pair of character sequences: one for input, and one for output.*

Namespaces

- namespace [std](#)

Functions

- `template<typename _CharT, typename _Traits >`
`streamsize std::__copy_streambufs_eof (basic_streambuf< _CharT, _Traits >`
`*, basic_streambuf< _CharT, _Traits > *, bool &)`
- `template<>`
`streamsize std::__copy_streambufs_eof (basic_streambuf< wchar_t > *__-`
`sbin, basic_streambuf< wchar_t > *__sbout, bool &__ineof)`
- `template<>`
`streamsize std::__copy_streambufs_eof (basic_streambuf< char > *__sbin,`
`basic_streambuf< char > *__sbout, bool &__ineof)`

6.321.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [streambuf](#).

6.322 streambuf.tcc File Reference

Namespaces

- namespace [std](#)

Functions

- `template<typename _CharT, typename _Traits >`
`streamsize std::__copy_streambufs (basic_streambuf< _CharT, _Traits > *_-`
`_sbin, basic_streambuf< _CharT, _Traits > *__sbout)`
- `template<typename _CharT, typename _Traits >`
`streamsize std::__copy_streambufs_eof (basic_streambuf< _CharT, _Traits >`
`*, basic_streambuf< _CharT, _Traits > *, bool &)`

6.322.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<streambuf>`.

Definition in file [streambuf.tcc](#).

6.323 `streambuf_iterator.h` File Reference

Classes

- class `std::istreambuf_iterator< _CharT, _Traits >`
Provides input iterator semantics for streambufs.
- class `std::ostreambuf_iterator< _CharT, _Traits >`
Provides output iterator semantics for streambufs.

Namespaces

- namespace `std`

Functions

- `template<bool _IsMove, typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type std::__copy_move_a2` (`_CharT * __first, _CharT * __last, ostreambuf_iterator< _CharT > __result`)
- `template<bool _IsMove, typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type std::__copy_move_a2` (`const _CharT * __first, const _CharT * __last, ostreambuf_iterator< _CharT > __result`)
- `template<bool _IsMove, typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type std::__copy_move_a2` (`istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, _CharT * __result`)
- `template<typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type std::copy` (`istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, ostreambuf_iterator< _CharT > __result`)
- `template<typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, istreambuf_iterator< _CharT > >::__type std::find` (`istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, const _CharT & __val`)
- `template<typename _CharT, typename _Traits >`
`bool std::operator!=` (`const istreambuf_iterator< _CharT, _Traits > & __a, const istreambuf_iterator< _CharT, _Traits > & __b`)
- `template<typename _CharT, typename _Traits >`
`bool std::operator==` (`const istreambuf_iterator< _CharT, _Traits > & __a, const istreambuf_iterator< _CharT, _Traits > & __b`)

6.323.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

Definition in file [streambuf_iterator.h](#).

6.324 string File Reference

6.324.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [string](#).

6.325 string File Reference

Classes

- class [__gnu_debug::basic_string<_CharT, _Traits, _Allocator>](#)
Class [std::basic_string](#) with safety/checking/debug instrumentation.

Namespaces

- namespace [__gnu_debug](#)

Typedefs

- typedef `basic_string< char >` [__gnu_debug::string](#)
- typedef `basic_string< wchar_t >` [__gnu_debug::wstring](#)

Functions

- `template<typename _CharT, typename _Traits, typename _Allocator>`
[std::basic_istream<_CharT, _Traits>](#) & [__gnu_debug::getline](#) ([std::basic_istream<_CharT, _Traits>](#) &__is, `basic_string<_CharT, _Traits, _Allocator>` &__str, `_CharT` __delim)
- `template<typename _CharT, typename _Traits, typename _Allocator>`
[std::basic_istream<_CharT, _Traits>](#) & [__gnu_debug::getline](#) ([std::basic_istream<_CharT, _Traits>](#) &__is, `basic_string<_CharT, _Traits, _Allocator>` &__str)

- [illegible]

- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool __gnu_debug::operator<= (const basic_string< _CharT, _Traits, _-`
`Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool __gnu_debug::operator== (const basic_string< _CharT, _Traits, _-`
`Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool __gnu_debug::operator== (const basic_string< _CharT, _Traits, _-`
`Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool __gnu_debug::operator== (const _CharT *__lhs, const basic_string< _-`
`CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool __gnu_debug::operator> (const basic_string< _CharT, _Traits, _-`
`Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool __gnu_debug::operator> (const _CharT *__lhs, const basic_string< _-`
`CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool __gnu_debug::operator> (const basic_string< _CharT, _Traits, _-`
`Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool __gnu_debug::operator>= (const _CharT *__lhs, const basic_string< _-`
`CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool __gnu_debug::operator>= (const basic_string< _CharT, _Traits, _-`
`Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool __gnu_debug::operator>= (const basic_string< _CharT, _Traits, _-`
`Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`std::basic_istream< _CharT, _Traits > & __gnu_debug::operator>>`
`(std::basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits,`
`_Allocator > &__str)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`void __gnu_debug::swap (basic_string< _CharT, _Traits, _Allocator > &__-`
`lhs, basic_string< _CharT, _Traits, _Allocator > &__rhs)`

6.325.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/string](#).

6.326 `string_conversions.h` File Reference

Namespaces

- namespace [__gnu_cxx](#)

Functions

- `template<typename _TRet , typename _Ret = _TRet, typename _CharT , typename... _Base>
_Ret __gnu_cxx::__stoa (_TRet(*__convf)(const _CharT *, _CharT **, _Base...), const char *__name, const _CharT *__str, std::size_t *__idx, _Base... _base)`
- `template<typename _String , typename _CharT = typename _String::value_type>
_String __gnu_cxx::__to_xstring (int(*__convf)(_CharT *, std::size_t, const _CharT *, __builtin_va_list), std::size_t __n, const _CharT *__fmt,...)`

6.326.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [string_conversions.h](#).

6.327 `stringfwd.h` File Reference

Namespaces

- namespace [std](#)

Typedefs

- `typedef basic_string< char > std::string`
- `typedef basic_string< char16_t > std::u16string`
- `typedef basic_string< char32_t > std::u32string`
- `typedef basic_string< wchar_t > std::wstring`

6.327.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<string>`.

Definition in file [stringfwd.h](#).

6.328 `strstream` File Reference

Namespaces

- namespace `std`

6.328.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<sstream>`.

Definition in file `strstream`.

6.329 `system_error` File Reference

Classes

- class `std::error_category`
error_category
- struct `std::error_code`
error_code
- struct `std::error_condition`
error_condition
- struct `std::hash< error_code >`
std::hash specialization for *error_code*.
- struct `std::is_error_code_enum< _Tp >`
is_error_code_enum
- struct `std::is_error_condition_enum< _Tp >`
is_error_condition_enum
- class `std::system_error`
Thrown to indicate error code of underlying system.

Namespaces

- namespace `std`

Functions

- `const error_category & std::generic_category () throw ()`
- `error_code std::make_error_code (errc __e)`
- `error_condition std::make_error_condition (errc __e)`
- `bool std::operator!= (const error_code &__lhs, const error_condition &__rhs)`
- `bool std::operator!= (const error_condition &__lhs, const error_condition &__rhs)`
- `bool std::operator!= (const error_code &__lhs, const error_code &__rhs)`
- `bool std::operator!= (const error_condition &__lhs, const error_code &__rhs)`
- `bool std::operator< (const error_code &__lhs, const error_code &__rhs)`
- `bool std::operator< (const error_condition &__lhs, const error_condition &__rhs)`
- `template<typename _CharT, typename _Traits >
basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, const error_code &__e)`
- `bool std::operator== (const error_condition &__lhs, const error_condition &__rhs)`
- `bool std::operator== (const error_code &__lhs, const error_condition &__rhs)`
- `bool std::operator== (const error_code &__lhs, const error_code &__rhs)`
- `bool std::operator== (const error_condition &__lhs, const error_code &__rhs)`
- `const error_category & std::system_category () throw ()`

Variables

- `error_code std::make_error_code (errc)`
- `error_condition std::make_error_condition (errc)`

6.329.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [system_error](#).

6.330 tag_and_trait.hpp File Reference

Classes

- `struct __gnu_pbds::associative_container_tag`
Basic associative-container.

- struct [__gnu_pbds::basic_hash_tag](#)
Basic hash.
- struct [__gnu_pbds::basic_tree_tag](#)
Basic tree.
- struct [__gnu_pbds::binary_heap_tag](#)
Binary-heap (array-based).
- struct [__gnu_pbds::binomial_heap_tag](#)
Binomial-heap.
- struct [__gnu_pbds::cc_hash_tag](#)
Collision-chaining hash.
- struct [__gnu_pbds::container_tag](#)
Base data structure tag.
- struct [__gnu_pbds::container_traits< Cntnr >](#)
container_traits
- struct [__gnu_pbds::gp_hash_tag](#)
General-probing hash.
- struct [__gnu_pbds::list_update_tag](#)
List-update.
- struct [__gnu_pbds::null_mapped_type](#)
A mapped-policy indicating that an associative container is a set.
- struct [__gnu_pbds::ov_tree_tag](#)
Ordered-vector tree.
- struct [__gnu_pbds::pairing_heap_tag](#)
Pairing-heap.
- struct [__gnu_pbds::pat_trie_tag](#)
PATRICIA trie.
- struct [__gnu_pbds::priority_queue_tag](#)
Basic priority-queue.

- struct [__gnu_pbds::rb_tree_tag](#)
Red-black tree.
- struct [__gnu_pbds::rc_binomial_heap_tag](#)
Redundant-counter binomial-heap.
- struct [__gnu_pbds::sequence_tag](#)
Basic sequence.
- struct [__gnu_pbds::splay_tree_tag](#)
Splay tree.
- struct [__gnu_pbds::string_tag](#)
Basic string container; inclusive of strings, ropes, etc.
- struct [__gnu_pbds::thin_heap_tag](#)
Thin heap.
- struct [__gnu_pbds::tree_tag](#)
tree.
- struct [__gnu_pbds::trie_tag](#)
trie.

Namespaces

- namespace [__gnu_pbds](#)

Typedefs

- typedef void [__gnu_pbds::trivial_iterator_difference_type](#)

6.330.1 Detailed Description

Contains tags and traits, e.g., ones describing underlying data structures.

Definition in file [tag_and_trait.hpp](#).

6.331 tags.h File Reference

Tags for compile-time selection. This file is a GNU parallel extension to the Standard C++ Library.

Classes

- struct [__gnu_parallel::balanced_quicksort_tag](#)
Forces parallel sorting using balanced quicksort at compile time.
- struct [__gnu_parallel::balanced_tag](#)
Recommends parallel execution using dynamic load-balancing at compile time.
- struct [__gnu_parallel::constant_size_blocks_tag](#)
Selects the constant block size variant for `std::find()`.
- struct [__gnu_parallel::default_parallel_tag](#)
Recommends parallel execution using the default parallel algorithm.
- struct [__gnu_parallel::equal_split_tag](#)
Selects the equal splitting variant for `std::find()`.
- struct [__gnu_parallel::exact_tag](#)
Forces parallel merging with exact splitting, at compile time.
- struct [__gnu_parallel::find_tag](#)
Base class for `std::find()` variants.
- struct [__gnu_parallel::growing_blocks_tag](#)
Selects the growing block size variant for `std::find()`.
- struct [__gnu_parallel::multiway_mergesort_exact_tag](#)
Forces parallel sorting using multiway mergesort with exact splitting at compile time.
- struct [__gnu_parallel::multiway_mergesort_sampling_tag](#)
Forces parallel sorting using multiway mergesort with splitting by sampling at compile time.
- struct [__gnu_parallel::multiway_mergesort_tag](#)
Forces parallel sorting using multiway mergesort at compile time.
- struct [__gnu_parallel::omp_loop_static_tag](#)
Recommends parallel execution using OpenMP static load-balancing at compile time.
- struct [__gnu_parallel::omp_loop_tag](#)
Recommends parallel execution using OpenMP dynamic load-balancing at compile time.

- struct [__gnu_parallel::parallel_tag](#)
Recommends parallel execution at compile time, optionally using a user-specified number of threads.
- struct [__gnu_parallel::quicksort_tag](#)
Forces parallel sorting using unbalanced quicksort at compile time.
- struct [__gnu_parallel::sampling_tag](#)
Forces parallel merging with exact splitting, at compile time.
- struct [__gnu_parallel::sequential_tag](#)
Forces sequential execution at compile time.
- struct [__gnu_parallel::unbalanced_tag](#)
Recommends parallel execution using static load-balancing at compile time.

Namespaces

- namespace [__gnu_parallel](#)

6.331.1 Detailed Description

Tags for compile-time selection. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [tags.h](#).

6.332 **tgmath.h** File Reference

6.332.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [tgmath.h](#).

6.333 **thread** File Reference

Classes

- struct [std::hash< thread::id >](#)
[std::hash](#) specialization for [thread::id](#).

- class [std::thread](#)
thread
- class [std::thread::id](#)
thread::id

Namespaces

- namespace [std](#)
- namespace [std::this_thread](#)

Functions

- `thread::id` [std::this_thread::get_id](#) ()
- `bool` **std::operator!=** (`thread::id` __x, `thread::id` __y)
- `template<class _CharT, class _Traits >`
`basic_ostream< _CharT, _Traits > & std::operator<<` (`basic_ostream< _CharT, _Traits > &__out`, `thread::id` __id)
- `bool` **std::operator**<= (`thread::id` __x, `thread::id` __y)
- `bool` **std::operator**> (`thread::id` __x, `thread::id` __y)
- `bool` **std::operator**>= (`thread::id` __x, `thread::id` __y)
- `template<typename _Rep, typename _Period >`
`void` [std::this_thread::sleep_for](#) (`const chrono::duration< _Rep, _Period > &__rtime`)
- `template<typename _Clock, typename _Duration >`
`void` [std::this_thread::sleep_until](#) (`const chrono::time_point< _Clock, _Duration > &__atime`)
- `void` **std::swap** (`thread &__x`, `thread &__y`)
- `void` [std::this_thread::yield](#) ()

6.333.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [thread](#).

6.334 `throw_allocator.h` File Reference

Classes

- struct [__gnu_cxx::annotate_base](#)

Base class for checking address and label information about allocations. Create a [`std::map`](#) between the allocated address (`void*`) and a datum for annotations, which are a pair of numbers corresponding to label and allocated size.

- struct [`__gnu_cxx::condition_base`](#)
Base struct for condition policy.
- struct [`__gnu_cxx::forced_error`](#)
Thrown by exception safety machinery.
- struct [`__gnu_cxx::limit_condition`](#)
Base class for incremental control and throw.
- struct [`__gnu_cxx::limit_condition::always_adjustor`](#)
Always enter the condition.
- struct [`__gnu_cxx::limit_condition::limit_adjustor`](#)
Enter the *n*th condition.
- struct [`__gnu_cxx::limit_condition::never_adjustor`](#)
Never enter the condition.
- struct [`__gnu_cxx::random_condition`](#)
Base class for random probability control and throw.
- struct [`__gnu_cxx::random_condition::always_adjustor`](#)
Always enter the condition.
- struct [`__gnu_cxx::random_condition::group_adjustor`](#)
Group condition.
- struct [`__gnu_cxx::random_condition::never_adjustor`](#)
Never enter the condition.
- class [`__gnu_cxx::throw_allocator_base< _Tp, _Cond >`](#)
Allocator class with logging and exception generation control. Intended to be used as an `allocator_type` in templated code.
Note: Deallocate not allowed to throw.
- struct [`__gnu_cxx::throw_allocator_limit< _Tp >`](#)
Allocator throwing via limit condition.
- struct [`__gnu_cxx::throw_allocator_random< _Tp >`](#)

Allocator throwing via random condition.

- struct `__gnu_cxx::throw_value_base< _Cond >`
Class with exception generation control. Intended to be used as a value_type in templated code.
- struct `__gnu_cxx::throw_value_limit`
Type throwing via limit condition.
- struct `__gnu_cxx::throw_value_random`
Type throwing via random condition.
- struct `std::hash< __gnu_cxx::throw_value_limit >`
Explicit specialization of `std::hash` for `__gnu_cxx::throw_value_limit`.
- struct `std::hash< __gnu_cxx::throw_value_random >`
Explicit specialization of `std::hash` for `__gnu_cxx::throw_value_limit`.

Namespaces

- namespace `__gnu_cxx`
- namespace `std`

Functions

- void `__gnu_cxx::__throw_forced_error ()`
- template<typename _Tp, typename _Cond >
bool `__gnu_cxx::operator!=` (const throw_allocator_base< _Tp, _Cond > &, const throw_allocator_base< _Tp, _Cond > &)
- template<typename _Cond >
throw_value_base< _Cond > `__gnu_cxx::operator*` (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- template<typename _Cond >
throw_value_base< _Cond > `__gnu_cxx::operator+` (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- template<typename _Cond >
throw_value_base< _Cond > `__gnu_cxx::operator-` (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- template<typename _Cond >
bool `__gnu_cxx::operator<` (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)

- [std::ostream](#) & `__gnu_cxx::operator<<` ([std::ostream](#) &os, const annotate_base &__b)
- `template<typename _Tp, typename _Cond >`
`bool __gnu_cxx::operator==` (const throw_allocator_base< _Tp, _Cond > &, const throw_allocator_base< _Tp, _Cond > &)
- `template<typename _Cond >`
`bool __gnu_cxx::operator==` (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- `template<typename _Cond >`
`void __gnu_cxx::swap` (throw_value_base< _Cond > &__a, throw_value_base< _Cond > &__b)

6.334.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Contains two exception-generating types (`throw_value`, `throw_allocator`) intended to be used as value and allocator types while testing exception safety in templated containers and algorithms. The allocator has additional log and debug features. The exception generated is of type `forced_exception_error`.

Definition in file [throw_allocator.h](#).

6.335 `time_members.h` File Reference

Namespaces

- namespace [std](#)

6.335.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

Definition in file [time_members.h](#).

6.336 `tree_policy.hpp` File Reference

Namespaces

- namespace [__gnu_pbds](#)

Defines

- #define **PB_DS_BASE_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**

6.336.1 Detailed Description

Contains tree-related policies.

Definition in file [tree_policy.hpp](#).

6.337 tree_trace_base.hpp File Reference

6.337.1 Detailed Description

Contains tree-related policies.

Definition in file [tree_trace_base.hpp](#).

6.338 trie_policy.hpp File Reference

Namespaces

- namespace [__gnu_pbds](#)

Defines

- #define **PB_DS_BASE_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**

6.338.1 Detailed Description

Contains trie-related policies.

Definition in file [trie_policy.hpp](#).

6.339 tuple File Reference

Classes

- struct `std::_Build_index_tuple<_Num>`
Builds an `_Index_tuple<0, 1, 2, ..., _Num-1>`.
- struct `std::_Index_tuple<_Indexes>`
- struct `std::_Tuple_impl<_Idx>`
- struct `std::_Tuple_impl<_Idx, _Head, _Tail...>`
- class `std::tuple<_Elements>`
tuple
- class `std::tuple<_T1>`
tuple (1-element).
- class `std::tuple<_T1, _T2>`
tuple (2-element), with construction and assignment from a pair.
- struct `std::tuple_element<0, tuple<_Head, _Tail...>>`
- struct `std::tuple_element<__i, tuple<_Head, _Tail...>>`
- struct `std::tuple_size<tuple<_Elements...>>`
class tuple_size

Namespaces

- namespace `std`

Functions

- `template<std::size_t __i, typename _Head, typename... _Tail>`
`__add_ref<_Head>::type std::__get_helper(_Tuple_impl<__i, _Head, _-`
`Tail...> &__t)`
- `template<std::size_t __i, typename _Head, typename... _Tail>`
`__add_c_ref<_Head>::type std::__get_helper(const _Tuple_impl<__i, _-`
`Head, _Tail...> &__t)`
- `template<typename... _TElements, std::size_t... _TIdx, typename... _UElements, std::size_t... _-`
`UIdx>`
`tuple<_TElements..., _UElements...> std::tuple_cat_helper(tuple<_-`
`TElements...> &&__t, const __index_holder<_TIdx...> &, tuple<_-`
`UElements...> &&__u, const __index_holder<_UIdx...> &)`

- `template<typename... _TElements, std::size_t... _TIdx, typename... _UElements, std::size_t... _UIdx>`
`tuple< _TElements..., _UElements...> std::__tuple_cat_helper (const tuple< _TElements...> &__t, const __index_holder< _TIdx...> &, const tuple< _UElements...> &__u, const __index_holder< _UIdx...> &)`
- `template<typename... _TElements, std::size_t... _TIdx, typename... _UElements, std::size_t... _UIdx>`
`tuple< _TElements..., _UElements...> std::__tuple_cat_helper (tuple< _TElements...> &&__t, const __index_holder< _TIdx...> &, const tuple< _UElements...> &__u, const __index_holder< _UIdx...> &)`
- `template<typename... _TElements, std::size_t... _TIdx, typename... _UElements, std::size_t... _UIdx>`
`tuple< _TElements..., _UElements...> std::__tuple_cat_helper (const tuple< _TElements...> &__t, const __index_holder< _TIdx...> &, tuple< _UElements...> &&__u, const __index_holder< _UIdx...> &)`
- `template<typename... _Elements>`
`tuple< _Elements &&...> std::forward_as_tuple (_Elements &&...__args)`
- `template<std::size_t _i, typename... _Elements>`
`__add_c_ref< typename tuple_element< __i, tuple< _Elements...> >::type >::type std::get (const tuple< _Elements...> &__t)`
- `template<std::size_t _i, typename... _Elements>`
`__add_ref< typename tuple_element< __i, tuple< _Elements...> >::type >::type std::get (tuple< _Elements...> &__t)`
- `template<typename... _Elements>`
`tuple< typename __decay_and_strip< _Elements >::__type...> std::make_tuple (_Elements &&...__args)`
- `template<typename... _TElements, typename... _UElements>`
`bool std::operator!= (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<typename... _TElements, typename... _UElements>`
`bool std::operator< (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<typename... _TElements, typename... _UElements>`
`bool std::operator<= (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<typename... _TElements, typename... _UElements>`
`bool std::operator== (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<typename... _TElements, typename... _UElements>`
`bool std::operator> (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<typename... _TElements, typename... _UElements>`
`bool std::operator>= (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`

- `template<typename... _Elements>`
`void std::swap (tuple< _Elements...> &__x, tuple< _Elements...> &__y)`
- `template<typename... _Elements>`
`tuple< _Elements &...> std::tie (_Elements &...__args)`
- `template<typename... _TElements, typename... _UElements>`
`tuple< _TElements..., _UElements...> std::tuple_cat (tuple< _TElements...>`
`&&__t, tuple< _UElements...> &&__u)`
- `template<typename... _TElements, typename... _UElements>`
`tuple< _TElements..., _UElements...> std::tuple_cat (const tuple< _-`
`TElements...> &__t, tuple< _UElements...> &&__u)`
- `template<typename... _TElements, typename... _UElements>`
`tuple< _TElements..., _UElements...> std::tuple_cat (const tuple< _-`
`TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<typename... _TElements, typename... _UElements>`
`tuple< _TElements..., _UElements...> std::tuple_cat (tuple< _TElements...>`
`&&__t, const tuple< _UElements...> &__u)`

Variables

- `const _Swallow_assign std::ignore`

6.339.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [tuple](#).

6.340 type_traits File Reference

Classes

- struct [std::__declval_protector< _Tp >](#)
declval
- struct [std::__is_member_pointer_helper< _Tp >](#)
is_member_pointer
- struct [std::add_const< _Tp >](#)
add_const
- struct [std::add_cv< _Tp >](#)
add_cv

- struct `std::add_lvalue_reference< _Tp >`
add_lvalue_reference
- struct `std::add_pointer< _Tp >`
add_pointer
- struct `std::add_rvalue_reference< _Tp >`
add_rvalue_reference
- struct `std::add_volatile< _Tp >`
add_volatile
- struct `std::aligned_storage< _Len, _Align >`
Alignment type.
- struct `std::alignment_of< _Tp >`
alignment_of
- struct `std::conditional< _Cond, _Iftrue, _Iffalse >`
conditional
- class `std::decay< _Tp >`
decay
- struct `std::enable_if< bool, _Tp >`
enable_if
- struct `std::extent< typename, _Uint >`
extent
- struct `std::has_nothrow_copy_assign< _Tp >`
has_nothrow_copy_assign
- struct `std::has_nothrow_copy_constructor< _Tp >`
has_nothrow_copy_constructor
- struct `std::has_nothrow_default_constructor< _Tp >`
has_nothrow_default_constructor
- struct `std::has_trivial_copy_assign< _Tp >`
has_trivial_copy_assign

- struct `std::has_trivial_copy_constructor< _Tp >`
has_trivial_copy_constructor
- struct `std::has_trivial_default_constructor< _Tp >`
has_trivial_default_constructor
- struct `std::has_trivial_destructor< _Tp >`
has_trivial_destructor
- struct `std::has_virtual_destructor< _Tp >`
has_virtual_destructor
- struct `std::integral_constant< _Tp, __v >`
integral_constant
- struct `std::is_abstract< _Tp >`
is_abstract
- struct `std::is_arithmetic< _Tp >`
is_arithmetic
- struct `std::is_array< typename >`
is_array
- struct `std::is_base_of< _Base, _Derived >`
is_base_of
- struct `std::is_class< _Tp >`
is_class
- struct `std::is_compound< _Tp >`
is_compound
- struct `std::is_const< typename >`
is_const
- struct `std::is_constructible< _Tp, _Args >`
is_constructible
- struct `std::is_convertible< _From, _To >`
is_convertible

- struct `std::is_empty< _Tp >`
is_empty
- struct `std::is_enum< _Tp >`
is_enum
- struct `std::is_explicitly_convertible< _From, _To >`
is_explicitly_convertible
- struct `std::is_floating_point< _Tp >`
is_floating_point
- struct `std::is_function< typename >`
is_function
- struct `std::is_fundamental< _Tp >`
is_fundamental
- struct `std::is_integral< _Tp >`
is_integral
- struct `std::is_literal_type< _Tp >`
is_literal_type
- struct `std::is_lvalue_reference< typename >`
is_lvalue_reference
- struct `std::is_member_function_pointer< _Tp >`
is_member_function_pointer
- struct `std::is_member_object_pointer< _Tp >`
is_member_object_pointer
- struct `std::is_nothrow_constructible< _Tp, _Args >`
is_nothrow_constructible
- struct `std::is_object< _Tp >`
is_object
- struct `std::is_pod< _Tp >`
is_pod

- struct `std::is_pointer< _Tp >`
is_pointer
- struct `std::is_polymorphic< _Tp >`
is_polymorphic
- struct `std::is_reference< _Tp >`
is_reference
- struct `std::is_rvalue_reference< typename >`
is_rvalue_reference
- struct `std::is_same< typename, typename >`
is_same
- struct `std::is_scalar< _Tp >`
is_scalar
- struct `std::is_signed< _Tp >`
is_signed
- struct `std::is_standard_layout< _Tp >`
is_standard_layout
- struct `std::is_trivial< _Tp >`
is_trivial
- struct `std::is_union< _Tp >`
is_union
- struct `std::is_unsigned< _Tp >`
is_unsigned
- struct `std::is_void< _Tp >`
is_void
- struct `std::is_volatile< typename >`
is_volatile
- struct `std::make_signed< _Tp >`
make_signed

- struct `std::make_unsigned< _Tp >`
make_unsigned
- struct `std::rank< typename >`
rank
- struct `std::remove_all_extents< _Tp >`
remove_all_extents
- struct `std::remove_const< _Tp >`
remove_const
- struct `std::remove_cv< _Tp >`
remove_cv
- struct `std::remove_extent< _Tp >`
remove_extent
- struct `std::remove_pointer< _Tp >`
remove_pointer
- struct `std::remove_reference< _Tp >`
remove_reference
- struct `std::remove_volatile< _Tp >`
remove_volatile

Namespaces

- namespace `std`

Defines

- `#define _DEFINE_SPEC(_Order, _Trait, _Type, _Value)`
- `#define _DEFINE_SPEC_0_HELPER`
- `#define _DEFINE_SPEC_1_HELPER`
- `#define _DEFINE_SPEC_2_HELPER`
- `#define _GLIBCXX_HAS_NESTED_TYPE(_NTYPE)`

Typedefs

- `typedef integral_constant< bool, false > std::false_type`
- `typedef integral_constant< bool, true > std::true_type`

Functions

- `template<typename _Tp >
add_rvalue_reference< _Tp >::type std::declval () noexcept`

6.340.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [type_traits](#).

6.341 `type_traits.h` File Reference

Namespaces

- namespace [__gnu_cxx](#)

Functions

- `template<typename _Type >
bool __gnu_cxx::__is_null_pointer (_Type *__ptr)`
- `template<typename _Type >
bool __gnu_cxx::__is_null_pointer (_Type)`

6.341.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [type_traits.h](#).

6.342 `type_utils.hpp` File Reference

Namespaces

- namespace [__gnu_pbds](#)

Defines

- `#define PB_DS_STATIC_ASSERT(UNIQUE, E)`

Typedefs

- `typedef std::tr1::integral_constant< int, 0 > __gnu_pbds::detail::false_type`
- `typedef std::tr1::integral_constant< int, 1 > __gnu_pbds::detail::true_type`

6.342.1 Detailed Description

Contains utilities for handling types. All of these classes are based on Modern C++ by Andrei Alexandrescu.

Definition in file [type_utils.hpp](#).

6.343 typeindex File Reference

Classes

- struct [std::hash< type_index >](#)
[std::hash](#) specialization for [type_index](#).
- struct [std::type_index](#)
The class [type_index](#) provides a simple wrapper for [type_info](#) which can be used as an index type in associative containers (23.6) and in unordered associative containers (23.7).

Namespaces

- namespace [std](#)

6.343.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [typeindex](#).

6.344 `typeinfo` File Reference

Classes

- class `std::bad_cast`
*Thrown during incorrect typecasting.
If you attempt an invalid `dynamic_cast` expression, an instance of this class (or something derived from this class) is thrown.*
- class `std::bad_typeid`
Thrown when a `NULL` pointer in a `typeid` expression is used.
- class `std::type_info`
Part of RTTI.

Namespaces

- namespace `std`

Defines

- `#define __GXX_MERGED_TYPEINFO_NAMES`
- `#define __GXX_TYPEINFO_EQUALITY_INLINE`

6.344.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [typeinfo](#).

6.345 `typelist.h` File Reference

Namespaces

- namespace `__gnu_cxx`
- namespace `__gnu_cxx::typelist`

Defines

- `#define _GLIBCXX_TYPELIST_CHAIN1(X0)`
- `#define _GLIBCXX_TYPELIST_CHAIN10(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9)`

- `#define _GLIBCXX_TYPELIST_CHAIN11(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10)`
- `#define _GLIBCXX_TYPELIST_CHAIN12(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11)`
- `#define _GLIBCXX_TYPELIST_CHAIN13(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12)`
- `#define _GLIBCXX_TYPELIST_CHAIN14(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13)`
- `#define _GLIBCXX_TYPELIST_CHAIN15(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14)`
- `#define _GLIBCXX_TYPELIST_CHAIN2(X0, X1)`
- `#define _GLIBCXX_TYPELIST_CHAIN3(X0, X1, X2)`
- `#define _GLIBCXX_TYPELIST_CHAIN4(X0, X1, X2, X3)`
- `#define _GLIBCXX_TYPELIST_CHAIN5(X0, X1, X2, X3, X4)`
- `#define _GLIBCXX_TYPELIST_CHAIN6(X0, X1, X2, X3, X4, X5)`
- `#define _GLIBCXX_TYPELIST_CHAIN7(X0, X1, X2, X3, X4, X5, X6)`
- `#define _GLIBCXX_TYPELIST_CHAIN8(X0, X1, X2, X3, X4, X5, X6, X7)`
- `#define _GLIBCXX_TYPELIST_CHAIN9(X0, X1, X2, X3, X4, X5, X6, X7, X8)`

Functions

- `template<typename Fn, typename Typelist >`
`void __gnu_cxx::typelist::apply (Fn &, Typelist)`
- `template<typename Fn, typename TypelistT, typename TypelistV >`
`void __gnu_cxx::typelist::apply_generator (Fn &fn, TypelistT, TypelistV)`
- `template<typename Fn, typename Typelist >`
`void __gnu_cxx::typelist::apply_generator (Fn &fn, Typelist)`
- `template<typename Gn, typename TypelistT, typename TypelistV >`
`void __gnu_cxx::typelist::apply_generator (Gn &, TypelistT, TypelistV)`
- `template<typename Gn, typename Typelist >`
`void __gnu_cxx::typelist::apply_generator (Gn &, Typelist)`

6.345.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Contains `typelist_chain` definitions. Typelists are an idea by Andrei Alexandrescu.

Definition in file [typelist.h](#).

6.346 types.h File Reference

Basic types and typedefs. This file is a GNU parallel extension to the Standard C++ Library.

Namespaces

- namespace [__gnu_parallel](#)

Typedefs

- typedef int64_t [__gnu_parallel::_CASable](#)
- typedef uint64_t [__gnu_parallel::_SequenceIndex](#)
- typedef uint16_t [__gnu_parallel::_ThreadIndex](#)

Enumerations

- enum [__gnu_parallel::_AlgorithmStrategy](#) { **heuristic**, **force_sequential**, **force_parallel** }
- enum [__gnu_parallel::_FindAlgorithm](#) { **GROWING_BLOCKS**, **CONSTANT_SIZE_BLOCKS**, **EQUAL_SPLIT** }
- enum [__gnu_parallel::_MultiwayMergeAlgorithm](#) { **LOSER_TREE** }
- enum [__gnu_parallel::_Parallelism](#) {
 [__gnu_parallel::sequential](#), [__gnu_parallel::parallel_unbalanced](#), [__gnu_parallel::parallel_balanced](#), [__gnu_parallel::parallel_omp_loop](#),
 [__gnu_parallel::parallel_omp_loop_static](#), [__gnu_parallel::parallel_taskqueue](#) }
- enum [__gnu_parallel::_PartialSumAlgorithm](#) { **RECURSIVE**, **LINEAR** }
- enum [__gnu_parallel::_SortAlgorithm](#) { **MWMS**, **QS**, **QS_BALANCED** }
- enum [__gnu_parallel::_SplittingAlgorithm](#) { **SAMPLING**, **EXACT** }

Variables

- static const int [__gnu_parallel::_CASable_bits](#)
- static const [_CASable](#) [__gnu_parallel::_CASable_mask](#)

6.346.1 Detailed Description

Basic types and typedefs. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [types.h](#).

6.347 types_traits.hpp File Reference

Namespaces

- namespace [__gnu_pbds](#)

6.347.1 Detailed Description

Contains a traits class of types used by containers.

Definition in file [types_traits.hpp](#).

6.348 unique_copy.h File Reference

Parallel implementations of `std::unique_copy()`. This file is a GNU parallel extension to the Standard C++ Library.

Namespaces

- namespace [__gnu_parallel](#)

Functions

- `template<typename _Iter, class _OutputIterator, class _BinaryPredicate >
_OutputIterator __gnu_parallel::__parallel_unique_copy (_Iter __first, _Iter __last, _OutputIterator __result, _BinaryPredicate __binary_pred)`
- `template<typename _Iter, class _OutputIterator >
_OutputIterator __gnu_parallel::__parallel_unique_copy (_Iter __first, _Iter __last, _OutputIterator __result)`

6.348.1 Detailed Description

Parallel implementations of `std::unique_copy()`. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [unique_copy.h](#).

6.349 unique_ptr.h File Reference

Classes

- struct `std::default_delete< _Tp >`

Primary template, *default_delete*.

- struct `std::default_delete< _Tp[]>`
Specialization, *default_delete*.
- struct `std::hash< unique_ptr< _Tp, _Dp > >`
std::hash specialization for *unique_ptr*.
- class `std::unique_ptr< _Tp, _Dp >`
20.7.12.2 *unique_ptr* for single objects.
- class `std::unique_ptr< _Tp[], _Dp >`
20.7.12.3 *unique_ptr* for array objects with a runtime length

Namespaces

- namespace `std`

Functions

- template<typename _Tp, typename _Dp, typename _Up, typename _Ep >
bool **std::operator!=** (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp, typename _Dp >
bool **std::operator!=** (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<typename _Tp, typename _Dp >
bool **std::operator!=** (nullptr_t, const unique_ptr< _Tp, _Dp > &__y)
- template<typename _Tp, typename _Dp, typename _Up, typename _Ep >
bool **std::operator<** (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp, typename _Dp, typename _Up, typename _Ep >
bool **std::operator<=** (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp, typename _Dp, typename _Up, typename _Ep >
bool **std::operator==** (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp, typename _Dp >
bool **std::operator==** (nullptr_t, const unique_ptr< _Tp, _Dp > &__y)
- template<typename _Tp, typename _Dp >
bool **std::operator==** (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<typename _Tp, typename _Dp, typename _Up, typename _Ep >
bool **std::operator>** (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)

- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >
bool std::operator>= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >
void std::swap (unique_ptr< _Tp, _Dp > &__x, unique_ptr< _Tp, _Dp > &__y)`

6.349.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

Definition in file [unique_ptr.h](#).

6.350 unordered_map File Reference

6.350.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [unordered_map](#).

6.351 unordered_map File Reference

Classes

- class [std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >](#)
Class [std::unordered_map](#) with safety/checking/debug instrumentation.
- class [std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >](#)
Class [std::unordered_multimap](#) with safety/checking/debug instrumentation.

Namespaces

- namespace [std](#)
- namespace [std::__debug](#)

Functions

- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`


```

bool std::__debug::operator!= (const unordered_map< _Key, _Tp, _Hash, _-
Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc
> &__y)
• template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc
>
bool std::__debug::operator!= (const unordered_multimap< _Key, _Tp, _-
Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash,
_Pred, _Alloc > &__y)
• template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc
>
bool std::__debug::operator== (const unordered_multimap< _Key, _Tp, _-
Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash,
_Pred, _Alloc > &__y)
• template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc
>
bool std::__debug::operator== (const unordered_map< _Key, _Tp, _Hash, _-
Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc
> &__y)
• template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc
>
void std::__debug::swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _-
Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__-
_y)
• template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc
>
void std::__debug::swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc
> &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)

```

6.351.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/unordered_map](#).

6.352 unordered_map File Reference

Classes

- class [std::__profile::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >](#)
Class [std::unordered_map](#) wrapper with performance instrumentation.
- class [std::__profile::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >](#)
Class [std::unordered_multimap](#) wrapper with performance instrumentation.

Namespaces

- namespace [std](#)
- namespace [std::__profile](#)

Defines

- `#define _GLIBCXX_BASE`
- `#define _GLIBCXX_BASE`
- `#define _GLIBCXX_STD_BASE`
- `#define _GLIBCXX_STD_BASE`

Functions

- `template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >`
`bool std::__profile::operator!= (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >`
`bool std::__profile::operator!= (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >`
`bool std::__profile::operator== (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >`
`bool std::__profile::operator== (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >`
`void std::__profile::swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >`
`void std::__profile::swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`

6.352.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/unordered_map](#).

6.353 unordered_map.h File Reference

Classes

- class [std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>](#)
A standard container composed of unique keys (containing at most one of each key value) that associates values of another type with the keys.
- class [std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>](#)
A standard container composed of equivalent keys (possibly containing multiple of each key value) that associates values of another type with the keys.

Namespaces

- namespace [std](#)

Functions

- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`bool std::operator!= (const __unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code> &__x, const __unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code> &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>`
`bool std::operator!= (const unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, const unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`bool std::operator!= (const __unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code> &__x, const __unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code> &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>`
`bool std::operator!= (const unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, const unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__y)`

- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`
`bool std::operator== (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc`
`> &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_`
`code>`
`bool std::operator== (const __unordered_map< _Key, _Tp, _Hash, _Pred, _`
`Alloc, __cache_hash_code > &__x, const __unordered_map< _Key, _Tp, _`
`Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_`
`code>`
`bool std::operator== (const __unordered_multimap< _Key, _Tp, _Hash, _`
`Pred, _Alloc, __cache_hash_code > &__x, const __unordered_multimap< _`
`Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`
`bool std::operator== (const unordered_multimap< _Key, _Tp, _Hash, _Pred,`
`_Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc`
`> &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_`
`code>`
`void std::swap (__unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc, __`
`cache_hash_code > &__x, __unordered_multimap< _Key, _Tp, _Hash, _Pred,`
`_Alloc, __cache_hash_code > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_`
`code>`
`void std::swap (__unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc, __`
`cache_hash_code > &__x, __unordered_map< _Key, _Tp, _Hash, _Pred, _`
`Alloc, __cache_hash_code > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`
`void std::swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x,`
`unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`
`void std::swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &_`
`__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`

6.353.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<unordered_map>`.

Definition in file [unordered_map.h](#).

6.354 unordered_set File Reference

6.354.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [unordered_set](#).

6.355 unordered_set File Reference

Classes

- class [std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >](#)
Class [std::unordered_multiset](#) with safety/checking/debug instrumentation.
- class [std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc >](#)
Class [std::unordered_set](#) with safety/checking/debug instrumentation.

Namespaces

- namespace [std](#)
- namespace [std::__debug](#)

Functions

- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >
bool std::__debug::operator!= (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >
bool std::__debug::operator!= (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >
bool std::__debug::operator== (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >
bool std::__debug::operator== (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >
void std::__debug::swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`

- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >
void std::__debug::swap (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`

6.355.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/unordered_set](#).

6.356 unordered_set File Reference

Classes

- class [std::__profile::unordered_multiset< _Value, _Hash, _Pred, _Alloc >](#)
Unordered_multiset wrapper with performance instrumentation.
- class [std::__profile::unordered_set< _Key, _Hash, _Pred, _Alloc >](#)
Unordered_set wrapper with performance instrumentation.

Namespaces

- namespace [std](#)
- namespace [std::__profile](#)

Defines

- `#define _GLIBCXX_BASE`
- `#define _GLIBCXX_BASE`
- `#define _GLIBCXX_STD_BASE`
- `#define _GLIBCXX_STD_BASE`

Functions

- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >
bool std::__profile::operator!= (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >
bool std::__profile::operator!= (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`

- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`bool std::__profile::operator==(const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`bool std::__profile::operator==(const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`void std::__profile::swap(unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`void std::__profile::swap(unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`

6.356.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/unordered_set](#).

6.357 unordered_set.h File Reference

Classes

- class [std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >](#)
A standard container composed of equivalent keys (possibly containing multiple of each key value) in which the elements' keys are the elements themselves.
- class [std::unordered_set< _Value, _Hash, _Pred, _Alloc >](#)
A standard container composed of unique keys (containing at most one of each key value) in which the elements' keys are the elements themselves.

Namespaces

- namespace [std](#)

Functions

- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`bool std::operator!=(const __unordered_set< _Value, _Hash, _Pred, _Alloc, __cache_hash_code > &__x, const __unordered_set< _Value, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)`

- `template<class _Value, class _Hash, class _Pred, class _Alloc >`
`bool std::operator!= (const unordered_set< _Value, _Hash, _Pred, _Alloc >`
`&__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`bool std::operator!= (const __unordered_multiset< _Value, _Hash, _Pred, _-`
`Alloc, __cache_hash_code > &__x, const __unordered_multiset< _Value, _-`
`Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`
`bool std::operator!= (const unordered_multiset< _Value, _Hash, _Pred, _Alloc`
`> &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`
`bool std::operator== (const unordered_set< _Value, _Hash, _Pred, _Alloc >`
`&__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`bool std::operator== (const __unordered_set< _Value, _Hash, _Pred, _Alloc,`
`__cache_hash_code > &__x, const __unordered_set< _Value, _Hash, _Pred,`
`_Alloc, __cache_hash_code > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`bool std::operator== (const __unordered_multiset< _Value, _Hash, _Pred, _-`
`Alloc, __cache_hash_code > &__x, const __unordered_multiset< _Value, _-`
`Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`
`bool std::operator== (const unordered_multiset< _Value, _Hash, _Pred, _-`
`Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &_-`
`__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`void std::swap (__unordered_multiset< _Value, _Hash, _Pred, _Alloc, __-`
`cache_hash_code > &__x, __unordered_multiset< _Value, _Hash, _Pred, _-`
`Alloc, __cache_hash_code > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`void std::swap (__unordered_set< _Value, _Hash, _Pred, _Alloc, __cache_-`
`hash_code > &__x, __unordered_set< _Value, _Hash, _Pred, _Alloc, __-`
`cache_hash_code > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`
`void std::swap (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x,`
`unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`
`void std::swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x,`
`unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`

6.357.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<unordered_set>`.

Definition in file [unordered_set.h](#).

6.358 utility File Reference

Namespaces

- namespace [std](#)

Functions

- `template<std::size_t _Int, class _Tp1, class _Tp2 >
tuple_element< _Int, std::pair< _Tp1, _Tp2 > >::type & std::get (std::pair< _Tp1, _Tp2 > &__in)`
- `template<std::size_t _Int, class _Tp1, class _Tp2 >
const tuple_element< _Int, std::pair< _Tp1, _Tp2 > >::type & std::get (const std::pair< _Tp1, _Tp2 > &__in)`

6.358.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [utility](#).

6.359 valarray File Reference

Classes

- class [std::valarray](#)< _Tp >
Smart array designed to support numeric processing.

Namespaces

- namespace [std](#)

Defines

- `#define _DEFINE_BINARY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_AUGMENTED_ASSIGNMENT(_Op, _Name)`
- `#define _DEFINE_VALARRAY_EXPR_AUGMENTED_ASSIGNMENT(_Op, _Name)`
- `#define _DEFINE_VALARRAY_UNARY_OPERATOR(_Op, _Name)`

Functions

- `template<class _Tp >`
`_Tp * std::begin (valarray< _Tp > &__va)`
- `template<class _Tp >`
`const _Tp * std::begin (const valarray< _Tp > &__va)`
- `template<class _Tp >`
`_Tp * std::end (valarray< _Tp > &__va)`
- `template<class _Tp >`
`const _Tp * std::end (const valarray< _Tp > &__va)`
- `template<typename _Tp >`
`_Expr< _BinClos< __not_equal_to, _ValArray, _Constant, _Tp, _Tp >, type-
name __fun< __not_equal_to, _Tp >::result_type > std::operator!= (const
valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __not_equal_to, _Constant, _ValArray, _Tp, _Tp >, type-
name __fun< __not_equal_to, _Tp >::result_type > std::operator!= (const _
Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __not_equal_to, _ValArray, _ValArray, _Tp, _Tp >, type-
name __fun< __not_equal_to, _Tp >::result_type > std::operator!= (const
valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __modulus, _ValArray, _ValArray, _Tp, _Tp >, typename
__fun< __modulus, _Tp >::result_type > std::operator% (const valarray< _
Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __modulus, _ValArray, _Constant, _Tp, _Tp >, typename
__fun< __modulus, _Tp >::result_type > std::operator% (const valarray< _
Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __modulus, _Constant, _ValArray, _Tp, _Tp >, typename
__fun< __modulus, _Tp >::result_type > std::operator% (const _Tp &__t,
const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_and, _ValArray, _Constant, _Tp, _Tp >, type-
name __fun< __bitwise_and, _Tp >::result_type > std::operator& (const
valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_and, _Constant, _ValArray, _Tp, _Tp >, type-
name __fun< __bitwise_and, _Tp >::result_type > std::operator& (const _Tp
&__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_and, _ValArray, _ValArray, _Tp, _Tp >, type-
name __fun< __bitwise_and, _Tp >::result_type > std::operator& (const
valarray< _Tp > &__v, const valarray< _Tp > &__w)`

- `template<typename _Tp >`
`_Expr< _BinClos< __logical_and, _ValArray, _Constant, _Tp, _Tp >, type-`
`name __fun< __logical_and, _Tp >::result_type > std::operator&& (const`
`valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_and, _Constant, _ValArray, _Tp, _Tp >, type-`
`name __fun< __logical_and, _Tp >::result_type > std::operator&& (const _`
`Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_and, _ValArray, _ValArray, _Tp, _Tp >, type-`
`name __fun< __logical_and, _Tp >::result_type > std::operator&& (const`
`valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __multiplies, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __multiplies, _Tp >::result_type > std::operator* (const valarray< _`
`Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __multiplies, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __multiplies, _Tp >::result_type > std::operator* (const _Tp &__t,`
`const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __multiplies, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __multiplies, _Tp >::result_type > std::operator* (const valarray< _`
`Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __plus, _ValArray, _Constant, _Tp, _Tp >, typename _`
`__fun< __plus, _Tp >::result_type > std::operator+ (const valarray< _Tp >`
`&__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __plus, _ValArray, _ValArray, _Tp, _Tp >, typename _`
`__fun< __plus, _Tp >::result_type > std::operator+ (const valarray< _Tp >`
`&__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __plus, _Constant, _ValArray, _Tp, _Tp >, typename _`
`__fun< __plus, _Tp >::result_type > std::operator+ (const _Tp &__t, const`
`valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __minus, _ValArray, _ValArray, _Tp, _Tp >, typename _`
`__fun< __minus, _Tp >::result_type > std::operator- (const valarray< _Tp >`
`&__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __minus, _ValArray, _Constant, _Tp, _Tp >, typename _`
`__fun< __minus, _Tp >::result_type > std::operator- (const valarray< _Tp >`
`&__v, const _Tp &__t)`

- `template<typename _Tp >`
`_Expr< _BinClos< __minus, _Constant, _ValArray, _Tp, _Tp >, typename _`
`_fun< __minus, _Tp >::result_type > std::operator- (const _Tp &__t, const`
`valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __divides, _ValArray, _ValArray, _Tp, _Tp >, typename _`
`_fun< __divides, _Tp >::result_type > std::operator/ (const valarray< _Tp >`
`&__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __divides, _ValArray, _Constant, _Tp, _Tp >, typename _`
`_fun< __divides, _Tp >::result_type > std::operator/ (const valarray< _Tp >`
`&__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __divides, _Constant, _ValArray, _Tp, _Tp >, typename _`
`_fun< __divides, _Tp >::result_type > std::operator/ (const _Tp &__t, const`
`valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __less, _Constant, _ValArray, _Tp, _Tp >, typename _`
`_fun< __less, _Tp >::result_type > std::operator< (const _Tp &__t, const`
`valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __less, _ValArray, _ValArray, _Tp, _Tp >, typename _`
`_fun< __less, _Tp >::result_type > std::operator< (const valarray< _Tp >`
`&__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __less, _ValArray, _Constant, _Tp, _Tp >, typename _`
`_fun< __less, _Tp >::result_type > std::operator< (const valarray< _Tp >`
`&__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __shift_left, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __shift_left, _Tp >::result_type > std::operator<< (const _Tp &__t,`
`const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __shift_left, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __shift_left, _Tp >::result_type > std::operator<< (const valarray<`
`_Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __shift_left, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __shift_left, _Tp >::result_type > std::operator<< (const valarray<`
`_Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __less_equal, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __less_equal, _Tp >::result_type > std::operator<= (const valarray<`
`_Tp > &__v, const valarray< _Tp > &__w)`

- `template<typename _Tp >`
`_Expr< _BinClos< __less_equal, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __less_equal, _Tp >::result_type > std::operator<= (const valarray<`
`_Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __less_equal, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __less_equal, _Tp >::result_type > std::operator<= (const _Tp &__t,`
`const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __equal_to, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __equal_to, _Tp >::result_type > std::operator== (const valarray< _`
`Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __equal_to, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __equal_to, _Tp >::result_type > std::operator== (const valarray< _`
`Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __equal_to, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __equal_to, _Tp >::result_type > std::operator== (const _Tp &__t,`
`const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater, _ValArray, _Constant, _Tp, _Tp >, typename _`
`__fun< __greater, _Tp >::result_type > std::operator> (const valarray< _Tp`
`> &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater, _ValArray, _ValArray, _Tp, _Tp >, typename _`
`__fun< __greater, _Tp >::result_type > std::operator> (const valarray< _Tp`
`> &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater, _Constant, _ValArray, _Tp, _Tp >, typename _`
`__fun< __greater, _Tp >::result_type > std::operator> (const _Tp &__t, const`
`valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater_equal, _ValArray, _ValArray, _Tp, _Tp >, type-`
`name __fun< __greater_equal, _Tp >::result_type > std::operator>= (const`
`valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater_equal, _Constant, _ValArray, _Tp, _Tp >, type-`
`name __fun< __greater_equal, _Tp >::result_type > std::operator>= (const`
`_Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater_equal, _ValArray, _Constant, _Tp, _Tp >, type-`
`name __fun< __greater_equal, _Tp >::result_type > std::operator>= (const`
`valarray< _Tp > &__v, const _Tp &__t)`

- `template<typename _Tp >`
`_Expr< _BinClos< __shift_right, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __shift_right, _Tp >::result_type > std::operator>> (const _Tp &__t,`
`const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __shift_right, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __shift_right, _Tp >::result_type > std::operator>> (const valarray<`
`_Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __shift_right, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __shift_right, _Tp >::result_type > std::operator>> (const valarray<`
`_Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_xor, _ValArray, _Constant, _Tp, _Tp >, type-`
`name __fun< __bitwise_xor, _Tp >::result_type > std::operator^ (const`
`valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_xor, _Constant, _ValArray, _Tp, _Tp >, type-`
`name __fun< __bitwise_xor, _Tp >::result_type > std::operator^ (const _Tp`
`&__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_xor, _ValArray, _ValArray, _Tp, _Tp >, type-`
`name __fun< __bitwise_xor, _Tp >::result_type > std::operator^ (const`
`valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_or, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __bitwise_or, _Tp >::result_type > std::operator| (const valarray< _`
`Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_or, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __bitwise_or, _Tp >::result_type > std::operator| (const valarray< _`
`Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_or, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __bitwise_or, _Tp >::result_type > std::operator| (const _Tp &__t,`
`const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_or, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __logical_or, _Tp >::result_type > std::operator|| (const valarray< _`
`Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_or, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __logical_or, _Tp >::result_type > std::operator|| (const valarray< _`
`Tp > &__v, const _Tp &__t)`

- `template<typename _Tp >
_Expr< _BinClos< __logical_or, _Constant, _ValArray, _Tp, _Tp >, typename
__fun< __logical_or, _Tp >::result_type > std::operator|| (const _Tp &__t,
const valarray< _Tp > &__v)`

6.359.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [valarray](#).

6.360 valarray_after.h File Reference

Namespaces

- namespace [std](#)

Defines

- `#define _DEFINE_EXPR_BINARY_FUNCTION(_Fun, _UFun)`
- `#define _DEFINE_EXPR_BINARY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_EXPR_UNARY_FUNCTION(_Name, _UName)`
- `#define _DEFINE_EXPR_UNARY_OPERATOR(_Op, _Name)`

Functions

- `template<class _Dom >
_Expr< _UnClos< _Abs, _Expr, _Dom >, typename _Dom::value_type >
std::abs (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >
_Expr< _UnClos< _Abs, _ValArray, _Tp >, _Tp > std::abs (const valarray<
_Tp > &__v)`
- `template<class _Dom >
_Expr< _UnClos< _Acos, _Expr, _Dom >, typename _Dom::value_type >
std::acos (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >
_Expr< _UnClos< _Acos, _ValArray, _Tp >, _Tp > std::acos (const valarray<
_Tp > &__v)`
- `template<class _Dom >
_Expr< _UnClos< _Asin, _Expr, _Dom >, typename _Dom::value_type >
std::asin (const _Expr< _Dom, typename _Dom::value_type > &__e)`

- `template<typename _Tp >`
`_Expr< _UnClos< _Asin, _ValArray, _Tp >, _Tp > std::asin (const valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Atan, _Expr, _Dom >, typename _Dom::value_type > std::atan (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Atan, _ValArray, _Tp >, _Tp > std::atan (const valarray< _Tp > &__v)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< _Atan2, _Expr, _Expr, _Dom1, _Dom2 >, typename _Dom1::value_type > std::atan2 (const _Expr< _Dom1, typename _Dom1::value_type > &__e1, const _Expr< _Dom2, typename _Dom2::value_type > &__e2)`
- `template<class _Dom >`
`_Expr< _BinClos< _Atan2, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _Dom::value_type > std::atan2 (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< _Atan2, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type > std::atan2 (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< _Atan2, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _Dom::value_type > std::atan2 (const _Expr< _Dom, typename _Dom::value_type > &__e, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< _Atan2, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type > std::atan2 (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Atan2, _ValArray, _ValArray, _Tp, _Tp >, _Tp > std::atan2 (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Atan2, _ValArray, _Constant, _Tp, _Tp >, _Tp > std::atan2 (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Atan2, _Constant, _ValArray, _Tp, _Tp >, _Tp > std::atan2 (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Cos, _Expr, _Dom >, typename _Dom::value_type > std::cos (const _Expr< _Dom, typename _Dom::value_type > &__e)`

- `template<typename _Tp >`
`_Expr< _UnClos< _Cos, _ValArray, _Tp >, _Tp > std::cos (const valarray<`
`_Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Cosh, _Expr, _Dom >, typename _Dom::value_type >`
`std::cosh (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Cosh, _ValArray, _Tp >, _Tp > std::cosh (const valarray<`
`_Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Exp, _Expr, _Dom >, typename _Dom::value_type >`
`std::exp (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Exp, _ValArray, _Tp >, _Tp > std::exp (const valarray<`
`_Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Log, _ValArray, _Tp >, _Tp > std::log (const valarray<`
`_Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Log, _Expr, _Dom >, typename _Dom::value_type >`
`std::log (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _UnClos< _Log10, _Expr, _Dom >, typename _Dom::value_type >`
`std::log10 (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Log10, _ValArray, _Tp >, _Tp > std::log10 (const`
`valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __not_equal_to, _Constant, _Expr, typename _-`
`Dom::value_type, _Dom >, typename __fun< __not_equal_to, typename _-`
`Dom::value_type >::result_type > std::operator!= (const typename _-`
`Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type >`
`&__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __not_equal_to, _Expr, _ValArray, _Dom, typename`
`_Dom::value_type >, typename __fun< __not_equal_to, typename _-`
`Dom::value_type >::result_type > std::operator!= (const _Expr< _Dom, type-`
`name _Dom::value_type > &__e, const valarray< typename _Dom::value_type`
`> &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __not_equal_to, _ValArray, _Expr, typename _-`
`Dom::value_type, _Dom >, typename __fun< __not_equal_to, typename _-`
`Dom::value_type >::result_type > std::operator!= (const valarray<`
`typename _Dom::value_type > &__v, const _Expr< _Dom, typename`
`_Dom::value_type > &__e)`

- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __not_equal_to, _Expr, _Expr, _Dom1, _Dom2 >, type-`
`name __fun< __not_equal_to, typename _Dom1::value_type >::result_type >`
`std::operator!= (const _Expr< _Dom1, typename _Dom1::value_type > &__v,`
`const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __not_equal_to, _Expr, _Constant, _Dom, typename`
`_Dom::value_type >, typename __fun< __not_equal_to, typename _`
`Dom::value_type >::result_type > std::operator!= (const _Expr< _Dom, type-`
`name _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __modulus, _Expr, _Constant, _Dom, typename _`
`Dom::value_type >, typename __fun< __modulus, typename _Dom::value-`
`type >::result_type > std::operator% (const _Expr< _Dom, typename _`
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __modulus, _Expr, _Expr, _Dom1, _Dom2 >, type-`
`name __fun< __modulus, typename _Dom1::value_type >::result_type >`
`std::operator% (const _Expr< _Dom1, typename _Dom1::value_type > &__`
`v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __modulus, _Constant, _Expr, typename _Dom::value-`
`type, _Dom >, typename __fun< __modulus, typename _Dom::value_type`
`>::result_type > std::operator% (const typename _Dom::value_type &__t,`
`const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __modulus, _Expr, _ValArray, _Dom, typename _`
`Dom::value_type >, typename __fun< __modulus, typename _Dom::value-`
`type >::result_type > std::operator% (const _Expr< _Dom, typename _`
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__`
`v)`
- `template<class _Dom >`
`_Expr< _BinClos< __modulus, _ValArray, _Expr, typename _Dom::value-`
`type, _Dom >, typename __fun< __modulus, typename _Dom::value_type`
`>::result_type > std::operator% (const valarray< typename _Dom::value-`
`type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_and, _Expr, _Constant, _Dom, typename`
`_Dom::value_type >, typename __fun< __bitwise_and, typename _`
`Dom::value_type >::result_type > std::operator& (const _Expr< _Dom,`
`typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_and, _Constant, _Expr, typename _Dom::value-`
`type, _Dom >, typename __fun< __bitwise_and, typename _Dom::value_type`

>::result_type > **std::operator&** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)

- template<class _Dom >
_Expr< _BinClos< __bitwise_and, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __bitwise_and, typename _Dom::value_type >::result_type > **std::operator&** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom >
_Expr< _BinClos< __bitwise_and, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __bitwise_and, typename _Dom::value_type >::result_type > **std::operator&** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom1, class _Dom2 >
_Expr< _BinClos< __bitwise_and, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __bitwise_and, typename _Dom1::value_type >::result_type > **std::operator&** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom1, class _Dom2 >
_Expr< _BinClos< __logical_and, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __logical_and, typename _Dom1::value_type >::result_type > **std::operator&&** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom >
_Expr< _BinClos< __logical_and, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __logical_and, typename _Dom::value_type >::result_type > **std::operator&&** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom >
_Expr< _BinClos< __logical_and, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __logical_and, typename _Dom::value_type >::result_type > **std::operator&&** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom >
_Expr< _BinClos< __logical_and, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __logical_and, typename _Dom::value_type >::result_type > **std::operator&&** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom >
_Expr< _BinClos< __logical_and, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __logical_and, typename _Dom::value_type >::result_type > **std::operator&&** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)

- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __multiplies, _Expr, _Expr, _Dom1, _Dom2 >, type-`
`name __fun< __multiplies, typename _Dom1::value_type >::result_type >`
`std::operator* (const _Expr< _Dom1, typename _Dom1::value_type > &__v,`
`const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __multiplies, _Expr, _Constant, _Dom, typename _-`
`Dom::value_type >, typename __fun< __multiplies, typename _Dom::value-`
`type >::result_type > std::operator* (const _Expr< _Dom, typename _-`
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __multiplies, _Constant, _Expr, typename _Dom::value-`
`type, _Dom >, typename __fun< __multiplies, typename _Dom::value_type`
`>::result_type > std::operator* (const typename _Dom::value_type &__t,`
`const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __multiplies, _Expr, _ValArray, _Dom, typename _-`
`Dom::value_type >, typename __fun< __multiplies, typename _Dom::value-`
`type >::result_type > std::operator* (const _Expr< _Dom, typename _-`
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &_-`
`__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __multiplies, _ValArray, _Expr, typename _Dom::value-`
`type, _Dom >, typename __fun< __multiplies, typename _Dom::value_type`
`>::result_type > std::operator* (const valarray< typename _Dom::value_type`
`> &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< __plus, _Expr, _ValArray, _Dom, typename _Dom::value-`
`type >, typename __fun< __plus, typename _Dom::value_type >::result_type`
`> std::operator+ (const _Expr< _Dom, typename _Dom::value_type > &__e,`
`const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __plus, _Expr, _Constant, _Dom, typename _Dom::value-`
`type >, typename __fun< __plus, typename _Dom::value_type >::result_type`
`> std::operator+ (const _Expr< _Dom, typename _Dom::value_type > &__v,`
`const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __plus, _ValArray, _Expr, typename _Dom::value_type, _-`
`Dom >, typename __fun< __plus, typename _Dom::value_type >::result_type`
`> std::operator+ (const valarray< typename _Dom::value_type > &__v, const`
`_Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __plus, _Expr, _Expr, _Dom1, _Dom2 >, typename _-`
`fun< __plus, typename _Dom1::value_type >::result_type > std::operator+`
`(const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr<`
`_Dom2, typename _Dom2::value_type > &__w)`

- `template<class _Dom >`
`_Expr< _BinClos< __plus, _Constant, _Expr, typename _Dom::value_type, _`
`Dom >, typename __fun< __plus, typename _Dom::value_type >::result_type`
`> std::operator+ (const typename _Dom::value_type &__t, const _Expr< _`
`Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __minus, _Constant, _Expr, typename _Dom::value_type,`
`_Dom >, typename __fun< __minus, typename _Dom::value_type >::result_`
`type > std::operator- (const typename _Dom::value_type &__t, const _Expr<`
`_Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __minus, _Expr, _ValArray, _Dom, typename _`
`Dom::value_type >, typename __fun< __minus, typename _Dom::value_`
`type >::result_type > std::operator- (const _Expr< _Dom, typename _`
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &_`
`__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __minus, _ValArray, _Expr, typename _Dom::value_type,`
`_Dom >, typename __fun< __minus, typename _Dom::value_type >::result_`
`type > std::operator- (const valarray< typename _Dom::value_type > &__v,`
`const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __minus, _Expr, _Expr, _Dom1, _Dom2 >, typename _`
`fun< __minus, typename _Dom1::value_type >::result_type > std::operator-`
`(const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr<`
`_Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __minus, _Expr, _Constant, _Dom, typename _`
`Dom::value_type >, typename __fun< __minus, typename _Dom::value_`
`type >::result_type > std::operator- (const _Expr< _Dom, typename _`
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __divides, _Expr, _Expr, _Dom1, _Dom2 >, typename _`
`fun< __divides, typename _Dom1::value_type >::result_type > std::operator/`
`(const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr<`
`_Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __divides, _Expr, _Constant, _Dom, typename _`
`Dom::value_type >, typename __fun< __divides, typename _Dom::value_`
`type >::result_type > std::operator/ (const _Expr< _Dom, typename _`
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __divides, _Constant, _Expr, typename _Dom::value_type,`
`_Dom >, typename __fun< __divides, typename _Dom::value_type >::result_`
`type > std::operator/ (const typename _Dom::value_type &__t, const _Expr<`
`_Dom, typename _Dom::value_type > &__v)`

- `template<class _Dom >`
`_Expr< _BinClos< __divides, _Expr, _ValArray, _Dom, typename _-`
`Dom::value_type >, typename __fun< __divides, typename _Dom::value_-`
`type >::result_type > std::operator/ (const _Expr< _Dom, typename _-`
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &_-`
`_v)`
- `template<class _Dom >`
`_Expr< _BinClos< __divides, _ValArray, _Expr, typename _Dom::value_type,`
`_Dom >, typename __fun< __divides, typename _Dom::value_type >::result_-`
`type > std::operator/ (const valarray< typename _Dom::value_type > &__v,`
`const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __less, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun<`
`__less, typename _Dom1::value_type >::result_type > std::operator< (const`
`_Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2,`
`typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __less, _Expr, _Constant, _Dom, typename _Dom::value_-`
`type >, typename __fun< __less, typename _Dom::value_type >::result_type`
`> std::operator< (const _Expr< _Dom, typename _Dom::value_type > &__v,`
`const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __less, _Constant, _Expr, typename _Dom::value_type, _-`
`Dom >, typename __fun< __less, typename _Dom::value_type >::result_type`
`> std::operator< (const typename _Dom::value_type &__t, const _Expr< _-`
`Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __less, _Expr, _ValArray, _Dom, typename _Dom::value_-`
`type >, typename __fun< __less, typename _Dom::value_type >::result_type`
`> std::operator< (const _Expr< _Dom, typename _Dom::value_type > &__e,`
`const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __less, _ValArray, _Expr, typename _Dom::value_type, _-`
`Dom >, typename __fun< __less, typename _Dom::value_type >::result_type`
`> std::operator< (const valarray< typename _Dom::value_type > &__v, const`
`_Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __shift_left, _Expr, _Expr, _Dom1, _Dom2 >, type-`
`name __fun< __shift_left, typename _Dom1::value_type >::result_type >`
`std::operator<< (const _Expr< _Dom1, typename _Dom1::value_type > &_-`
`_v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_left, _Expr, _Constant, _Dom, typename _-`
`Dom::value_type >, typename __fun< __shift_left, typename _Dom::value_-`
`type >::result_type > std::operator<< (const _Expr< _Dom, typename _-`
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`

- `template<class _Dom >`
`_Expr< _BinClos< __shift_left, _Constant, _Expr, typename _Dom::value_`
`type, _Dom >, typename __fun< __shift_left, typename _Dom::value_type`
`>::result_type > std::operator<< (const typename _Dom::value_type &__t,`
`const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_left, _Expr, _ValArray, _Dom, typename _`
`Dom::value_type >, typename __fun< __shift_left, typename _Dom::value_`
`type >::result_type > std::operator<< (const _Expr< _Dom, typename _`
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &_`
`__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_left, _ValArray, _Expr, typename _Dom::value_`
`type, _Dom >, typename __fun< __shift_left, typename _Dom::value_type`
`>::result_type > std::operator<< (const valarray< typename _Dom::value_`
`type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __less_equal, _Expr, _Expr, _Dom1, _Dom2 >, type-`
`name __fun< __less_equal, typename _Dom1::value_type >::result_type >`
`std::operator<= (const _Expr< _Dom1, typename _Dom1::value_type > &_`
`__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __less_equal, _Expr, _Constant, _Dom, typename _`
`Dom::value_type >, typename __fun< __less_equal, typename _Dom::value_`
`type >::result_type > std::operator<= (const _Expr< _Dom, typename _`
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __less_equal, _Expr, _ValArray, _Dom, typename _`
`Dom::value_type >, typename __fun< __less_equal, typename _Dom::value_`
`type >::result_type > std::operator<= (const _Expr< _Dom, typename _`
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &_`
`__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __less_equal, _Constant, _Expr, typename _Dom::value_`
`type, _Dom >, typename __fun< __less_equal, typename _Dom::value_type`
`>::result_type > std::operator<= (const typename _Dom::value_type &__t,`
`const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __less_equal, _ValArray, _Expr, typename _Dom::value_`
`type, _Dom >, typename __fun< __less_equal, typename _Dom::value_type`
`>::result_type > std::operator<= (const valarray< typename _Dom::value_`
`type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< __equal_to, _Expr, _ValArray, _Dom, typename _`
`Dom::value_type >, typename __fun< __equal_to, typename _Dom::value_`

```
type >::result_type > std::operator== (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
```

- template<class _Dom >
_Expr< _BinClos< __equal_to, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __equal_to, typename _Dom::value_type >::result_type > **std::operator==** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom >
_Expr< _BinClos< __equal_to, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __equal_to, typename _Dom::value_type >::result_type > **std::operator==** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom1, class _Dom2 >
_Expr< _BinClos< __equal_to, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __equal_to, typename _Dom1::value_type >::result_type > **std::operator==** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom >
_Expr< _BinClos< __equal_to, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __equal_to, typename _Dom::value_type >::result_type > **std::operator==** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom >
_Expr< _BinClos< __greater, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __greater, typename _Dom::value_type >::result_type > **std::operator>** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom >
_Expr< _BinClos< __greater, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __greater, typename _Dom::value_type >::result_type > **std::operator>** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom >
_Expr< _BinClos< __greater, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __greater, typename _Dom::value_type >::result_type > **std::operator>** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom1, class _Dom2 >
_Expr< _BinClos< __greater, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __greater, typename _Dom1::value_type >::result_type > **std::operator>** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)

- `template<class _Dom >`
`_Expr< _BinClos< __greater, _Constant, _Expr, typename _Dom::value_type,`
`_Dom >, typename __fun< __greater, typename _Dom::value_type >::result_`
`type > std::operator> (const typename _Dom::value_type &__t, const _Expr<`
`_Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __greater_equal, _Expr, _ValArray, _Dom, typename`
`_Dom::value_type >, typename __fun< __greater_equal, typename _`
`Dom::value_type >::result_type > std::operator>= (const _Expr< _Dom,`
`typename _Dom::value_type > &__e, const valarray< typename _Dom::value_`
`type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __greater_equal, _Constant, _Expr, typename _`
`Dom::value_type, _Dom >, typename __fun< __greater_equal, typename`
`_Dom::value_type >::result_type > std::operator>= (const typename _`
`Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type >`
`&__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __greater_equal, _ValArray, _Expr, typename _`
`Dom::value_type, _Dom >, typename __fun< __greater_equal, typename _`
`Dom::value_type >::result_type > std::operator>= (const valarray< typename`
`_Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type`
`> &__e)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __greater_equal, _Expr, _Expr, _Dom1, _Dom2 >, type-`
`name __fun< __greater_equal, typename _Dom1::value_type >::result_type >`
`std::operator>= (const _Expr< _Dom1, typename _Dom1::value_type > &_`
`__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __greater_equal, _Expr, _Constant, _Dom, typename`
`_Dom::value_type >, typename __fun< __greater_equal, typename _`
`Dom::value_type >::result_type > std::operator>= (const _Expr< _Dom,`
`typename _Dom::value_type > &__v, const typename _Dom::value_type &_`
`__t)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __shift_right, _Expr, _Expr, _Dom1, _Dom2 >, type-`
`name __fun< __shift_right, typename _Dom1::value_type >::result_type >`
`std::operator>> (const _Expr< _Dom1, typename _Dom1::value_type > &_`
`__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_right, _Expr, _Constant, _Dom, typename _`
`Dom::value_type >, typename __fun< __shift_right, typename _Dom::value_`
`type >::result_type > std::operator>> (const _Expr< _Dom, typename _`
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`

- `template<class _Dom >`
`_Expr< _BinClos< __shift_right, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __shift_right, typename _Dom::value_type >::result_type > std::operator>>` (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- `template<class _Dom >`
`_Expr< _BinClos< __shift_right, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __shift_right, typename _Dom::value_type >::result_type > std::operator>>` (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- `template<class _Dom >`
`_Expr< _BinClos< __shift_right, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __shift_right, typename _Dom::value_type >::result_type > std::operator>>` (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_xor, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __bitwise_xor, typename _Dom::value_type >::result_type > std::operator^` (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_xor, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __bitwise_xor, typename _Dom::value_type >::result_type > std::operator^` (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __bitwise_xor, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __bitwise_xor, typename _Dom1::value_type >::result_type > std::operator^` (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_xor, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __bitwise_xor, typename _Dom::value_type >::result_type > std::operator^` (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_xor, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __bitwise_xor, typename _Dom::value_type >::result_type > std::operator^` (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __bitwise_or, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __bitwise_or, typename _Dom1::value_type >::result_type >`

std::operator| (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)

- template<class _Dom >
_Expr< _BinClos< __bitwise_or, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __bitwise_or, typename _Dom::value_type >::result_type > **std::operator|** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom >
_Expr< _BinClos< __bitwise_or, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __bitwise_or, typename _Dom::value_type >::result_type > **std::operator|** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom >
_Expr< _BinClos< __bitwise_or, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __bitwise_or, typename _Dom::value_type >::result_type > **std::operator|** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom >
_Expr< _BinClos< __bitwise_or, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __bitwise_or, typename _Dom::value_type >::result_type > **std::operator|** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom >
_Expr< _BinClos< __logical_or, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __logical_or, typename _Dom::value_type >::result_type > **std::operator||** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom >
_Expr< _BinClos< __logical_or, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __logical_or, typename _Dom::value_type >::result_type > **std::operator||** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom >
_Expr< _BinClos< __logical_or, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __logical_or, typename _Dom::value_type >::result_type > **std::operator||** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom >
_Expr< _BinClos< __logical_or, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __logical_or, typename _Dom::value_type >::result_type > **std::operator||** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)

- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __logical_or, _Expr, _Expr, _Dom1, _Dom2 >, type-`
`name __fun< __logical_or, typename _Dom1::value_type >::result_type >`
`std::operator||` (const `_Expr< _Dom1, typename _Dom1::value_type > &__v,`
`const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< _Pow, _ValArray, _Expr, typename _Dom::value_type, _`
`Dom >, typename _Dom::value_type > std::pow` (const `valarray< typename`
`_Dom::valarray > &__v, const _Expr< _Dom, typename _Dom::value_type >
&__e)`
- `template<class _Dom >`
`_Expr< _BinClos< _Pow, _Constant, _Expr, typename _Dom::value_type,`
`_Dom >, typename _Dom::value_type > std::pow` (const `typename _`
`Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type >
&__e)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Pow, _ValArray, _Constant, _Tp, _Tp >, _Tp > std::pow`
`(const valarray< _Tp > &__v, const _Tp &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< _Pow, _Expr, _Constant, _Dom, typename _Dom::value _`
`type >, typename _Dom::value_type > std::pow` (const `_Expr< _Dom, type-`
`name _Dom::value_type > &__e, const typename _Dom::value_type &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Pow, _ValArray, _ValArray, _Tp, _Tp >, _Tp > std::pow`
`(const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< _Pow, _Expr, _ValArray, _Dom, typename _Dom::value _`
`type >, typename _Dom::value_type > std::pow` (const `_Expr< _Dom, type-`
`name _Dom::value_type > &__e, const valarray< typename _Dom::value_type
> &__v)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< _Pow, _Expr, _Expr, _Dom1, _Dom2 >, type-`
`name _Dom1::value_type > std::pow` (const `_Expr< _Dom1, typename _`
`Dom1::value_type > &__e1, const _Expr< _Dom2, typename _Dom2::value _
type > &__e2)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Pow, _Constant, _ValArray, _Tp, _Tp >, _Tp > std::pow`
`(const _Tp &__t, const valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Sin, _Expr, _Dom >, typename _Dom::value_type >`
`std::sin` (const `_Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Sin, _ValArray, _Tp >, _Tp > std::sin` (const `valarray<`
`_Tp > &__v)`

- `template<class _Dom >`
`_Expr< _UnClos< _Sinh, _Expr, _Dom >, typename _Dom::value_type >`
`std::sinh (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Sinh, _ValArray, _Tp >, _Tp > std::sinh (const valarray<`
`_Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Sqrt, _ValArray, _Tp >, _Tp > std::sqrt (const valarray<`
`_Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Sqrt, _Expr, _Dom >, typename _Dom::value_type >`
`std::sqrt (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Tan, _ValArray, _Tp >, _Tp > std::tan (const valarray<`
`_Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Tan, _Expr, _Dom >, typename _Dom::value_type >`
`std::tan (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _UnClos< _Tanh, _Expr, _Dom >, typename _Dom::value_type >`
`std::tanh (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Tanh, _ValArray, _Tp >, _Tp > std::tanh (const valarray<`
`_Tp > &__v)`

6.360.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

Definition in file [valarray_after.h](#).

6.361 valarray_array.h File Reference

Namespaces

- namespace [std](#)

Defines

- `#define _DEFINE_ARRAY_FUNCTION(_Op, _Name)`

Functions

- `template<typename _Tp >`
`void std::__valarray_copy (const _Tp *__restrict __a, size_t __n, _Tp *__-`
`restrict __b)`
- `template<typename _Tp >`
`void std::__valarray_copy (const _Tp *__restrict __src, size_t __n, size_t`
`__s1, _Tp *__restrict __dst, size_t __s2)`
- `template<typename _Tp >`
`void std::__valarray_copy (const _Tp *__restrict __a, const size_t *__-`
`restrict __i, _Tp *__restrict __b, size_t __n)`
- `template<typename _Tp >`
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, size_t __s1, _-`
`Array< _Tp > __b, size_t __s2)`
- `template<typename _Tp >`
`void std::__valarray_copy (_Array< _Tp > __a, _Array< size_t > __i, _-`
`Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp >`
`__b, _Array< size_t > __i)`
- `template<typename _Tp >`
`void std::__valarray_copy (_Array< _Tp > __src, size_t __n, _Array< size_t`
`> __i, _Array< _Tp > __dst, _Array< size_t > __j)`
- `template<typename _Tp >`
`void std::__valarray_copy (const _Tp *__restrict __a, size_t __n, _Tp *__-`
`restrict __b, const size_t *__restrict __i)`
- `template<typename _Tp >`
`void std::__valarray_copy (const _Tp *__restrict __src, size_t __n, const`
`size_t *__restrict __i, _Tp *__restrict __dst, const size_t *__restrict __j)`
- `template<typename _Tp >`
`void std::__valarray_copy (const _Tp *__restrict __a, size_t __n, size_t __s,`
`_Tp *__restrict __b)`
- `template<typename _Tp >`
`void std::__valarray_copy (const _Tp *__restrict __a, _Tp *__restrict __-`
`b, size_t __n, size_t __s)`
- `template<typename _Tp >`
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp >`
`__b)`
- `template<typename _Tp >`
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, size_t __s, _-`
`Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::__valarray_copy (_Array< _Tp > __a, _Array< _Tp > __b, size_t`
`__n, size_t __s)`

- `template<typename _Tp >`
`void std::__valarray_copy_construct (const _Tp *__restrict__ __a, const`
`size_t *__restrict__ __i, _Tp *__restrict__ __o, size_t __n)`
- `template<typename _Tp >`
`void std::__valarray_copy_construct (const _Tp *__restrict__ __a, size_t __n,`
`size_t __s, _Tp *__restrict__ __o)`
- `template<typename _Tp >`
`void std::__valarray_copy_construct (_Array< _Tp > __a, _Array< size_t >`
`__i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void std::__valarray_copy_construct (_Array< _Tp > __a, size_t __n, size_t`
`__s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::__valarray_copy_construct (const _Tp *__b, const _Tp *__e, _Tp`
`*__restrict__ __o)`
- `template<typename _Tp >`
`void std::__valarray_default_construct (_Tp *__b, _Tp *__e)`
- `template<typename _Tp >`
`void std::__valarray_destroy_elements (_Tp *__b, _Tp *__e)`
- `template<typename _Tp >`
`void std::__valarray_fill (_Tp *__restrict__ __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`
`void std::__valarray_fill (_Tp *__restrict__ __a, size_t __n, size_t __s, const`
`_Tp &__t)`
- `template<typename _Tp >`
`void std::__valarray_fill (_Tp *__restrict__ __a, const size_t *__restrict__ __i,`
`size_t __n, const _Tp &__t)`
- `template<typename _Tp >`
`void std::__valarray_fill (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`
`void std::__valarray_fill (_Array< _Tp > __a, size_t __n, size_t __s, const`
`_Tp &__t)`
- `template<typename _Tp >`
`void std::__valarray_fill (_Array< _Tp > __a, _Array< size_t > __i, size_t`
`__n, const _Tp &__t)`
- `template<typename _Tp >`
`void std::__valarray_fill_construct (_Tp *__b, _Tp *__e, const _Tp __t)`
- `void * std::__valarray_get_memory (size_t __n)`
- `template<typename _Tp >`
`_Tp *__restrict__ std::__valarray_get_storage (size_t __n)`
- `template<typename _Ta >`
`_Ta::value_type std::__valarray_max (const _Ta &__a)`
- `template<typename _Ta >`
`_Ta::value_type std::__valarray_min (const _Ta &__a)`

- `template<typename _Tp >`
`_Tp std::__valarray_product (const _Tp *__f, const _Tp *__l)`
- `void std::__valarray_release_memory (void *__p)`
- `template<typename _Tp >`
`_Tp std::__valarray_sum (const _Tp *__f, const _Tp *__l)`
- `template<typename _Tp, class _Dom >`
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::__Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp >`
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n,`
`const _Tp &__t)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n,`
`_Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, const _`
`Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n,`
`size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< _`
`Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __s,`
`const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, _Array<`
`size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, _Array<`
`size_t > __i, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, _Array<`
`bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n,`
`_Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n,`
`_Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n,`
`const _Tp &__t)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n,`
`_Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, const _`
`Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n,`
`size_t __s, _Array< _Tp > __b)`

- `template<typename _Tp >`
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array<`
`_Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __s,`
`const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array<`
`size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array<`
`bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array<`
`size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array<`
`bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n,`
`_Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n,`
`_Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __n,`
`const _Tp &__t)`
- `template<typename _Tp >`
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __n, _`
`Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__divides (_Array< _Tp > __a, const _Expr<`
`_Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __n,`
`size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::Array_augmented__divides (_Array< _Tp > __a, _Array< _Tp`
`> __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __s,`
`const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__divides (_Array< _Tp > __a, _Array< size_t`
`> __i, _Array< _Tp > __b, size_t __n)`

- `template<typename _Tp >`
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__divides (_Array< _Tp > __a, _Array< size_t`
`> __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__divides (_Array< _Tp > __a, _Array< bool`
`> __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__divides (_Array< _Tp > __a, _Array< bool`
`> __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< bool`
`> __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, const`
`_Tp &__t)`
- `template<typename _Tp >`
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< size_t`
`> __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __s, const`
`_Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< size_t`
`> __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< _Tp >`
`__b, size_t __n, size_t __s)`
- `template<typename _Tp >`
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n,`
`size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__minus (_Array< _Tp > __a, const _Expr<`
`_Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp >`
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, _-`
`Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< bool`
`> __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n,`
`const _Tp &__t)`
- `template<typename _Tp >`
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n,`
`_Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__modulus (_Array< _Tp > __a, const _Expr<`
`_Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< _Tp`
`> __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __s,`
`const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array<`
`size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array<`
`size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< bool`
`> __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n,`
`_Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< bool`
`> __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n,`
`_Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n,`
`size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::Array_augmented__multiplies (_Array< _Tp > __a, _Array< _-`
`Tp > __b, size_t __n, size_t __s)`

- `template<typename _Tp >`
`void std::Array_augmented__multiplies (_Array< _Tp > __a, _Array<`
`size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __n,`
`_Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__multiplies (_Array< _Tp > __a, _Array<`
`size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__multiplies (_Array< _Tp > __a, _Array<`
`bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __n,`
`_Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __n,`
`size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__multiplies (_Array< _Tp > __a, const _`
`Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__multiplies (_Array< _Tp > __a, _Array<`
`bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __s,`
`const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __n,`
`const _Tp &__t)`
- `template<typename _Tp >`
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __n,`
`_Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::Array_augmented__plus (_Array< _Tp > __a, _Array< bool >`
`__m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__plus (_Array< _Tp > __a, size_t __s, const`
`_Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__plus (_Array< _Tp > __a, size_t __n, const`
`_Tp &__t)`
- `template<typename _Tp >`
`void std::Array_augmented__plus (_Array< _Tp > __a, size_t __n, _`
`Array< _Tp > __b)`

- `template<typename _Tp >`
`void std::Array_augmented__plus (_Array< _Tp > __a, _Array< _Tp >`
`__b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__plus (_Array< _Tp > __a, const Expr< _`
`Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__plus (_Array< _Tp > __a, _Array< size_t >`
`__i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__plus (_Array< _Tp > __a, _Array< size_t >`
`__i, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__plus (_Array< _Tp > __a, size_t __n, _`
`Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`
`void std::Array_augmented__plus (_Array< _Tp > __a, size_t __n, _`
`Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`
`void std::Array_augmented__plus (_Array< _Tp > __a, size_t __n, size_t`
`__s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__plus (_Array< _Tp > __a, _Array< bool >`
`__m, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_left (_Array< _Tp > __a, size_t __n,`
`const _Tp &__t)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__shift_left (_Array< _Tp > __a, const _`
`Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__shift_left (_Array< _Tp > __a, _Array<`
`bool > __m, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__shift_left (_Array< _Tp > __a, size_t __s,`
`const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_left (_Array< _Tp > __a, _Array<`
`bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_left (_Array< _Tp > __a, size_t __n,`
`size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_left (_Array< _Tp > __a, size_t __n,`
`_Array< _Tp > __b)`

- `template<typename _Tp >`
`void std::Array_augmented__shift_left (_Array< _Tp > __a, _Array<`
`size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_left (_Array< _Tp > __a, _Array< _Tp`
`> __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__shift_left (_Array< _Tp > __a, _Array<`
`size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_left (_Array< _Tp > __a, size_t __n,`
`_Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_left (_Array< _Tp > __a, size_t __n,`
`_Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__shift_right (_Array< _Tp > __a, _Array<`
`bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_right (_Array< _Tp > __a, _Array<`
`_Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __n,`
`_Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_right (_Array< _Tp > __a, _Array<`
`bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__shift_right (_Array< _Tp > __a, _Array<`
`size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __n,`
`_Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__shift_right (_Array< _Tp > __a, const _`
`Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __n,`
`_Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_right (_Array< _Tp > __a, _Array<`
`size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __n,`
`size_t __s, _Array< _Tp > __b)`

- `template<typename _Tp, class _Dom>`
`void std::__Array_augmented__shift_right (_Array< _Tp > __a, size_t __s,`
`const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`
`void std::__Array_augmented__shift_right (_Array< _Tp > __a, size_t __n,`
`const _Tp &__t)`

6.361.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

Definition in file [valarray_array.h](#).

6.362 valarray_array.tcc File Reference

Namespaces

- namespace [std](#)

Functions

- `template<typename _Tp>`
`void std::__valarray_copy (_Array< _Tp > __a, _Array< bool > __m, _-`
`Array< _Tp > __b, size_t __n)`
- `template<typename _Tp>`
`void std::__valarray_copy (_Array< _Tp > __a, _Array< bool > __m, size_t`
`__n, _Array< _Tp > __b, _Array< bool > __k)`
- `template<typename _Tp>`
`void std::__valarray_copy (_Array< _Tp > __e, _Array< size_t > __f, size_t`
`__n, _Array< _Tp > __a, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom>`
`void std::__valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _-`
`Array< _Tp > __a, _Array< bool > __m)`
- `template<typename _Tp, class _Dom>`
`void std::__valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _-`
`Array< _Tp > __a)`
- `template<typename _Tp, class _Dom>`
`void std::__valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _-`
`Array< _Tp > __a, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom>`
`void std::__valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _-`
`Array< _Tp > __a, size_t __s)`

- `template<typename _Tp >`
`void std::__valarray_copy` (`_Array< _Tp > __a`, `size_t __n`, `_Array< _Tp > __b`, `_Array< bool > __m`)
- `template<typename _Tp, class _Dom >`
`void std::__valarray_copy_construct` (`const _Expr< _Dom, _Tp > &__e`, `size_t __n`, `_Array< _Tp > __a`)
- `template<typename _Tp >`
`void std::__valarray_copy_construct` (`_Array< _Tp > __a`, `_Array< bool > __m`, `_Array< _Tp > __b`, `size_t __n`)
- `template<typename _Tp >`
`void std::__valarray_fill` (`_Array< _Tp > __a`, `size_t __n`, `_Array< bool > __m`, `const _Tp &__t`)

6.362.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

Definition in file [valarray_array.tcc](#).

6.363 `valarray_before.h` File Reference

Namespaces

- namespace [std](#)

6.363.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

Definition in file [valarray_before.h](#).

6.364 `vector` File Reference

6.364.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [vector](#).

6.365 vector File Reference

Classes

- class `std::__debug::vector< _Tp, _Allocator >`
Class `std::vector` with safety/checking/debug instrumentation.
- struct `std::hash< __debug::vector< bool, _Alloc > >`
`std::hash` specialization for `vector<bool>`.

Namespaces

- namespace `std`
- namespace `std::__debug`

Functions

- `template<typename _Tp, typename _Alloc >`
`bool std::__debug::operator!= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool std::__debug::operator< (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool std::__debug::operator<= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool std::__debug::operator== (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool std::__debug::operator> (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool std::__debug::operator>= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`void std::__debug::swap (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &__rhs)`

6.365.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/vector](#).

6.366 vector File Reference

Classes

- struct [std::hash< __profile::vector< bool, _Alloc > >](#)
std::hash specialization for vector<bool>.

Namespaces

- namespace [std](#)
- namespace [std::__profile](#)

Functions

- `template<typename _Tp, typename _Alloc >`
`bool std::__profile::operator!= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool std::__profile::operator< (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool std::__profile::operator<= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool std::__profile::operator== (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool std::__profile::operator> (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool std::__profile::operator>= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`void std::__profile::swap (vector< _Tp, _Alloc > &&__lhs, vector< _Tp, _Alloc > &&__rhs)`
- `template<typename _Tp, typename _Alloc >`
`void std::__profile::swap (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &&__rhs)`
- `template<typename _Tp, typename _Alloc >`
`void std::__profile::swap (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &&__rhs)`

6.366.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/vector](#).

6.367 **vector.tcc** File Reference

Namespaces

- namespace [std](#)

6.367.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<vector>`.

Definition in file [vector.tcc](#).

6.368 **vstring.h** File Reference

Classes

- class [__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>](#)
Template class [__versa_string](#).
Data structure managing sequences of characters and character-like objects.

Namespaces

- namespace [__gnu_cxx](#)
- namespace [std](#)

Functions

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > & __is, __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base> & __str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`

```
basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str)
```

- `template<typename _CharT, typename _Traits, typename _Alloc, template<typename, typename, typename> class _Base>`
`bool __gnu_cxx::operator!= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template<typename, typename, typename> class _Base>`
`bool __gnu_cxx::operator!= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template<typename, typename, typename> class _Base>`
`bool __gnu_cxx::operator!= (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template<typename, typename, typename> class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template<typename, typename, typename> class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (_versa_string< _CharT, _Traits, _Alloc, _Base > &&__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template<typename, typename, typename> class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template<typename, typename, typename> class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const _CharT *__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template<typename, typename, typename> class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template<typename, typename, typename> class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, _CharT __rhs)`

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (_`
`CharT __lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (_`
`__versa_string< _CharT, _Traits, _Alloc, _Base > &&__lhs, const _CharT *__`
`rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (_`
`CharT __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (_`
`__versa_string< _CharT, _Traits, _Alloc, _Base > &&__lhs, const __versa_`
`string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (__`
`versa_string< _CharT, _Traits, _Alloc, _Base > &&__lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+`
`(const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, __versa_`
`string< _CharT, _Traits, _Alloc, _Base > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool __gnu_cxx::operator< (const __versa_string< _CharT, _Traits, _Alloc, _`
`Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__`
`rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool __gnu_cxx::operator< (const __versa_string< _CharT, _Traits, _Alloc, _`
`Base > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool __gnu_cxx::operator< (const _CharT *__lhs, const __versa_string< _`
`CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _`
`CharT, _Traits > &__os, const __gnu_cxx::__versa_string< _CharT, _Traits,`
`_Alloc, _Base > &__str)`

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool __gnu_cxx::operator<= (const __versa_string< _CharT, _Traits, _Alloc,`
`_Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__`
`_rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,`
`typename > class _Base>`
`bool __gnu_cxx::operator<= (const __versa_string< _CharT, _Traits, _Alloc,`
`_Base > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,`
`typename > class _Base>`
`bool __gnu_cxx::operator<= (const _CharT *__lhs, const __versa_string< _`
`CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,`
`typename > class _Base>`
`bool __gnu_cxx::operator== (const __versa_string< _CharT, _Traits, _Alloc, _`
`Base > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,`
`typename > class _Base>`
`bool __gnu_cxx::operator== (const __versa_string< _CharT, _Traits, _Alloc,`
`_Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__`
`_rhs)`
- `template<typename _CharT, template< typename, typename, typename > class _Base>`
`__enable_if< std::__is_char< _CharT >::__value, bool >::__type __gnu -`
`cxx::operator== (const __versa_string< _CharT, std::char_traits< _CharT > ,`
`std::allocator< _CharT > , _Base > &__lhs, const __versa_string< _CharT,`
`std::char_traits< _CharT > , std::allocator< _CharT > , _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,`
`typename > class _Base>`
`bool __gnu_cxx::operator== (const _CharT *__lhs, const __versa_string< _`
`CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,`
`typename > class _Base>`
`bool __gnu_cxx::operator> (const __versa_string< _CharT, _Traits, _Alloc, _`
`Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__`
`rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,`
`typename > class _Base>`
`bool __gnu_cxx::operator> (const __versa_string< _CharT, _Traits, _Alloc, _`
`Base > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,`
`typename > class _Base>`
`bool __gnu_cxx::operator> (const _CharT *__lhs, const __versa_string< _`
`CharT, _Traits, _Alloc, _Base > &__rhs)`

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool __gnu_cxx::operator>= (const __versa_string< _CharT, _Traits, _Alloc,`
`_Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__`
`_rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,`
`typename > class _Base>`
`bool __gnu_cxx::operator>= (const __versa_string< _CharT, _Traits, _Alloc,`
`_Base > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,`
`typename > class _Base>`
`bool __gnu_cxx::operator>= (const _CharT *__lhs, const __versa_string< _`
`CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,`
`typename > class _Base>`
`basic_istream< _CharT, _Traits > &std::operator>> (basic_istream< _CharT,`
`_Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base`
`> &__str)`
- `double __gnu_cxx::stod (const __vstring &__str, std::size_t *__idx=0)`
- `float __gnu_cxx::stof (const __vstring &__str, std::size_t *__idx=0)`
- `int __gnu_cxx::stoi (const __vstring &__str, std::size_t *__idx=0, int __`
`base=10)`
- `long __gnu_cxx::stol (const __vstring &__str, std::size_t *__idx=0, int __`
`base=10)`
- `long double __gnu_cxx::stold (const __vstring &__str, std::size_t *__idx=0)`
- `long long __gnu_cxx::stoll (const __vstring &__str, std::size_t *__idx=0, int`
`__base=10)`
- `unsigned long __gnu_cxx::stoul (const __vstring &__str, std::size_t *__idx=0,`
`int __base=10)`
- `unsigned long long __gnu_cxx::stoull (const __vstring &__str, std::size_t *__`
`idx, int __base=10)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,`
`typename > class _Base>`
`void __gnu_cxx::swap (__versa_string< _CharT, _Traits, _Alloc, _Base > &__`
`_lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `__vstring __gnu_cxx::to_string (long __val)`
- `__vstring __gnu_cxx::to_string (double __val)`
- `__vstring __gnu_cxx::to_string (long long __val)`
- `__vstring __gnu_cxx::to_string (unsigned __val)`
- `__vstring __gnu_cxx::to_string (long double __val)`
- `__vstring __gnu_cxx::to_string (int __val)`
- `__vstring __gnu_cxx::to_string (unsigned long long __val)`
- `__vstring __gnu_cxx::to_string (float __val)`
- `__vstring __gnu_cxx::to_string (unsigned long __val)`

- `__wvstring __gnu_cxx::to_wstring` (double `__val`)
- `__wvstring __gnu_cxx::to_wstring` (long double `__val`)
- `__wvstring __gnu_cxx::to_wstring` (long long `__val`)
- `__wvstring __gnu_cxx::to_wstring` (float `__val`)
- `__wvstring __gnu_cxx::to_wstring` (int `__val`)
- `__wvstring __gnu_cxx::to_wstring` (unsigned `__val`)
- `__wvstring __gnu_cxx::to_wstring` (long `__val`)
- `__wvstring __gnu_cxx::to_wstring` (unsigned long long `__val`)
- `__wvstring __gnu_cxx::to_wstring` (unsigned long `__val`)

6.368.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [vstring.h](#).

6.369 `vstring.tcc` File Reference

Namespaces

- namespace [__gnu_cxx](#)
- namespace [std](#)

Functions

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > & __is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & __str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (_CharT __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, _CharT __rhs)`

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const`
`__versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__-`
`rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const`
`__versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_-`
`string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT,`
`_Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base`
`> &__str)`

6.369.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/vstring.h>`.

Definition in file `vstring.tcc`.

6.370 `vstring_fwd.h` File Reference

Namespaces

- namespace `__gnu_cxx`

Typedefs

- `typedef __versa_string< char, std::char_traits< char >, std::allocator< char >, __rc_string_base > __gnu_cxx::__rc_string`
- `typedef __vstring __gnu_cxx::__sso_string`
- `typedef __versa_string< char16_t, std::char_traits< char16_t >, std::allocator< char16_t >, __rc_string_base > __gnu_cxx::__u16rc_string`
- `typedef __u16vstring __gnu_cxx::__u16sso_string`
- `typedef __versa_string< char16_t > __gnu_cxx::__u16vstring`
- `typedef __versa_string< char32_t, std::char_traits< char32_t >, std::allocator< char32_t >, __rc_string_base > __gnu_cxx::__u32rc_string`
- `typedef __u32vstring __gnu_cxx::__u32sso_string`
- `typedef __versa_string< char32_t > __gnu_cxx::__u32vstring`
- `typedef __versa_string< char > __gnu_cxx::__vstring`

- typedef `__versa_string< wchar_t, std::char_traits< wchar_t >, std::allocator< wchar_t >, __rc_string_base > __gnu_cxx::__wrc_string`
- typedef `__wvstring __gnu_cxx::__wsso_string`
- typedef `__versa_string< wchar_t > __gnu_cxx::__wvstring`

6.370.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/vstring.h>`.

Definition in file [vstring_fwd.h](#).

6.371 `vstring_util.h` File Reference

Namespaces

- namespace [__gnu_cxx](#)

6.371.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/vstring.h>`.

Definition in file [vstring_util.h](#).

6.372 `workstealing.h` File Reference

Parallelization of embarrassingly parallel execution by means of work-stealing.

Classes

- struct [__gnu_parallel::__Job< _DifferenceTp >](#)
One __job for a certain thread.

Namespaces

- namespace [__gnu_parallel](#)

Defines

- `#define _GLIBCXX_JOB_VOLATILE`

Functions

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>
>
_Op __gnu_parallel::__for_each_template_random_access_workstealing (_-
RAIter __begin, _RAIter __end, _Op __op, _Fu &__f, _Red __r, _Result __base,
_Result &__output, typename std::iterator_traits< _RAIter >::difference_type
__bound)`

6.372.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of work-stealing. Work stealing is described in

R. D. Blumofe and C. E. Leiserson. Scheduling multithreaded computations by work stealing. *Journal of the ACM*, 46(5):720–748, 1999.

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [workstealing.h](#).

Index

- ~_LoserTreeBase
 - __gnu_parallel::_LoserTreeBase, 1239
- ~_RestrictedBoundedConcurrentQueue
 - __gnu_parallel::_RestrictedBoundedConcurrentQueue, 1273
- ~_Safe_sequence_base
 - __gnu_debug::_Safe_sequence_base, 1103
- ~__versa_string
 - __gnu_cxx::__versa_string, 844
- ~auto_ptr
 - std::auto_ptr, 1600
- ~basic_filebuf
 - std::basic_filebuf, 1620
- ~basic_fstream
 - std::basic_fstream, 1665
- ~basic_ifstream
 - std::basic_ifstream, 1736
- ~basic_ios
 - std::basic_ios, 1793
- ~basic_iostream
 - std::basic_iostream, 1833
- ~basic_istream
 - std::basic_istream, 1901
- ~basic_istreamream
 - std::basic_istreamream, 1962
- ~basic_ofstream
 - std::basic_ofstream, 2020
- ~basic_ostream
 - std::basic_ostream, 2069
- ~basic_ostringstream
 - std::basic_ostringstream, 2119
- ~basic_regex
 - std::basic_regex, 2161
- ~basic_streambuf
 - std::basic_streambuf, 2174
- ~basic_string
 - std::basic_string, 2206
- ~basic_stringstream
 - std::basic_stringstream, 2305
- ~collate
 - std::collate, 2445
- ~ctype
 - std::ctype< char >, 2487
 - std::ctype< wchar_t >, 2503
- ~deque
 - std::deque, 2565
- ~facet
 - std::locale::facet, 2872
- ~forward_list
 - std::forward_list, 2632
- ~gslice
 - numeric_arrays, 94
- ~ios_base
 - std::ios_base, 2745
- ~locale
 - std::locale, 2864
- ~match_results
 - std::match_results, 2918
- ~messages
 - std::messages, 2936
- ~money_get
 - std::money_get, 2949
- ~money_put
 - std::money_put, 2954
- ~moneypunct
 - std::moneypunct, 2962
- ~num_get
 - std::num_get, 3046
- ~num_put
 - std::num_put, 3066
- ~numpunct
 - std::numpunct, 3114
- ~sentry
 - std::basic_ostream::sentry, 2106
- ~stdio_filebuf
 - __gnu_cxx::stdio_filebuf, 1002
- ~temporary_buffer
 - __gnu_cxx::temporary_buffer, 1064
- ~time_get
 - std::time_get, 3299
- ~time_put

- std::time_put, 3322
- ~type_info
 - std::type_info, 3337
- ~vector
 - std::vector, 3377
- /mnt/share/src/ Directory Reference, 362
- /mnt/share/src/gcc.git-trunk/ Directory Reference, 347
- /mnt/share/src/gcc.git-trunk/libstdc++-v3/ Directory Reference, 354
- /mnt/share/src/gcc.git-trunk/libstdc++-v3/doc/ Directory Reference, 343
- /mnt/share/src/gcc.git-trunk/libstdc++-v3/doc/doxygen/ Directory Reference, 344
- /mnt/share/src/gcc.git-trunk/libstdc++-v3/libsupc++/ Directory Reference, 355
- __gnu_parallel
 - parallel_balanced, 431
 - parallel_omp_loop, 431
 - parallel_omp_loop_static, 431
 - parallel_taskqueue, 431
 - parallel_unbalanced, 431
 - sequential, 431
- _AlgorithmStrategy
 - __gnu_parallel, 430
- _BALLOC_ALIGN_BYTES
 - bitmap_allocator.h, 3470
- _BinIndex
 - __gnu_parallel, 429
- _Bit_scan_forward
 - __gnu_cxx, 388
- _CASable
 - __gnu_parallel, 429
- _CASable_bits
 - __gnu_parallel, 480
- _CASable_mask
 - __gnu_parallel, 480
- _Construct
 - std, 656
- _DRandomShufflingGlobalData
 - __gnu_parallel::-
 - DRandomShufflingGlobalData, 1208
- _Destroy
 - std, 656
- _FindAlgorithm
 - __gnu_parallel, 430
- _Find_first
 - SGIextensions, 15
- _Find_next
 - SGIextensions, 15
- _GLIBCXX_ASSERTIONS
 - compiletime_settings.h, 3500
- _GLIBCXX_ATOMIC_PROPERTY
 - atomics, 217
- _GLIBCXX_BAL_QUICKSORT
 - features.h, 3546
- _GLIBCXX_CALL
 - compiletime_settings.h, 3501
- _GLIBCXX_DEBUG_VERIFY
 - macros.h, 3610
- _GLIBCXX_DEQUEUE_BUF_SIZE
 - stl_deque.h, 3751
- _GLIBCXX_FIND_CONSTANT_-SIZE_BLOCKS
 - features.h, 3546
- _GLIBCXX_FIND_EQUAL_SPLIT
 - features.h, 3547
- _GLIBCXX_FIND_GROWING_-BLOCKS
 - features.h, 3547
- _GLIBCXX_HAS_NESTED_TYPE
 - metaprogramming, 300
- _GLIBCXX_MERGESORT
 - features.h, 3547
- _GLIBCXX_PARALLEL_CONDITION
 - settings.h, 3719
- _GLIBCXX_PARALLEL_LENGTH
 - multiway_merge.h, 3630
- _GLIBCXX_QUICKSORT
 - features.h, 3547
- _GLIBCXX_RANDOM_SHUFFLE_-CONSIDER_L1
 - compiletime_settings.h, 3501
- _GLIBCXX_RANDOM_SHUFFLE_-CONSIDER_TLB
 - compiletime_settings.h, 3501
- _GLIBCXX_SCALE_DOWN_FPU
 - compiletime_settings.h, 3502

-
- `_GLIBCXX_TREE_DYNAMIC_-BALANCING`
features.h, 3548
 - `_GLIBCXX_TREE_FULL_COPY`
features.h, 3548
 - `_GLIBCXX_TREE_INITIAL_-SPLITTING`
features.h, 3548
 - `_GLIBCXX_VERBOSE_LEVEL`
compiletime_settings.h, 3502
 - `_GLIBCXX_VOLATILE`
partition.h, 3654
queue.h, 3676
 - `_GuardedIterator`
__gnu_parallel::_GuardedIterator, 1217
 - `_LoserTreeBase`
__gnu_parallel::_LoserTreeBase, 1239
 - `_M_allocate_and_copy`
std::vector, 3378
 - `_M_allocate_single_object`
__gnu_cxx::bitmap_allocator, 919
 - `_M_attach`
__gnu_debug::_Safe_iterator, 1083
__gnu_debug::_Safe_iterator_base, 1094
__gnu_debug::_Safe_sequence, 1098
__gnu_debug::_Safe_sequence_base, 1103
__gnu_debug::basic_string, 1114
std::__debug::deque, 1411
std::__debug::forward_list, 1417
std::__debug::list, 1424
std::__debug::map, 1430
std::__debug::multimap, 1436
std::__debug::multiset, 1442
std::__debug::set, 1449
std::__debug::unordered_map, 1454
std::__debug::unordered_multimap, 1460
std::__debug::unordered_multiset, 1465
std::__debug::unordered_set, 1471
std::__debug::vector, 1478
 - `_M_attach_single`
__gnu_debug::_Safe_iterator, 1083
__gnu_debug::_Safe_iterator_base, 1094
__gnu_debug::_Safe_sequence, 1098
__gnu_debug::_Safe_sequence_base, 1103
__gnu_debug::basic_string, 1114
std::__debug::deque, 1411
std::__debug::forward_list, 1417
std::__debug::list, 1424
std::__debug::map, 1430
std::__debug::multimap, 1436
std::__debug::multiset, 1442
std::__debug::set, 1449
std::__debug::unordered_map, 1454
std::__debug::unordered_multimap, 1460
std::__debug::unordered_multiset, 1465
std::__debug::unordered_set, 1471
std::__debug::vector, 1478
 - `_M_attached_to`
__gnu_debug::_Safe_iterator, 1083
__gnu_debug::_Safe_iterator_base, 1094
 - `_M_before_dereferenceable`
__gnu_debug::_Safe_iterator, 1083
 - `_M_begin`
__gnu_parallel::_Piece, 1260
 - `_M_bin_proc`
__gnu_parallel::_DRandomShufflingGlobalData, 1209
 - `_M_bins_begin`
__gnu_parallel::_DRSSorterPU, 1211
 - `_M_buf`
__gnu_cxx::enc_filebuf, 950
__gnu_cxx::stdio_filebuf, 1026
std::basic_filebuf, 1643
 - `_M_buf_locale`
__gnu_cxx::enc_filebuf, 950
__gnu_cxx::stdio_filebuf, 1027
-

-
- __gnu_cxx::stdio_sync_filebuf, 1057
 - std::basic_filebuf, 1643
 - std::basic_streambuf, 2193
 - std::basic_stringbuf, 2286
 - _M_buf_size
 - __gnu_cxx::enc_filebuf, 950
 - __gnu_cxx::stdio_filebuf, 1027
 - std::basic_filebuf, 1643
 - _M_can_compare
 - __gnu_debug::_Safe_iterator, 1084
 - __gnu_debug::_Safe_iterator_base, 1094
 - _M_clear
 - __gnu_cxx::free_list, 960
 - _M_comp
 - __gnu_parallel::_LoserTree, 1233
 - __gnu_parallel::_LoserTree< false, _Tp, _Compare >, 1237
 - __gnu_parallel::_LoserTreeBase, 1241
 - _M_const_iterators
 - __gnu_debug::_Safe_sequence, 1100
 - __gnu_debug::_Safe_sequence_base, 1105
 - __gnu_debug::basic_string, 1160
 - std::__debug::deque, 1413
 - std::__debug::forward_list, 1419
 - std::__debug::list, 1426
 - std::__debug::map, 1432
 - std::__debug::multimap, 1438
 - std::__debug::multiset, 1444
 - std::__debug::set, 1451
 - std::__debug::unordered_map, 1457
 - std::__debug::unordered_multimap, 1462
 - std::__debug::unordered_multiset, 1467
 - std::__debug::unordered_set, 1473
 - std::__debug::vector, 1480
 - _M_create_node
 - std::list, 2840
 - _M_create_pback
 - __gnu_cxx::enc_filebuf, 932
 - __gnu_cxx::stdio_filebuf, 1003
 - std::basic_filebuf, 1620
 - _M_data
 - std::_List_node, 1547
 - _M_deallocate_single_object
 - __gnu_cxx::bitmap_allocator, 920
 - _M_dereferenceable
 - __gnu_debug::_Safe_iterator, 1084
 - _M_destroy_pback
 - __gnu_cxx::enc_filebuf, 932
 - __gnu_cxx::stdio_filebuf, 1003
 - std::basic_filebuf, 1620
 - _M_detach
 - __gnu_debug::_Safe_iterator, 1084
 - __gnu_debug::_Safe_iterator_base, 1094
 - __gnu_debug::_Safe_sequence, 1098
 - __gnu_debug::_Safe_sequence_base, 1103
 - __gnu_debug::basic_string, 1114
 - std::__debug::deque, 1411
 - std::__debug::forward_list, 1417
 - std::__debug::list, 1424
 - std::__debug::map, 1430
 - std::__debug::multimap, 1436
 - std::__debug::multiset, 1442
 - std::__debug::set, 1449
 - std::__debug::unordered_map, 1455
 - std::__debug::unordered_multimap, 1460
 - std::__debug::unordered_multiset, 1465
 - std::__debug::unordered_set, 1471
 - std::__debug::vector, 1478
 - _M_detach_all
 - __gnu_debug::_Safe_sequence, 1098
 - __gnu_debug::_Safe_sequence_base, 1103
 - __gnu_debug::basic_string, 1114
 - std::__debug::deque, 1411
 - std::__debug::forward_list, 1417
 - std::__debug::list, 1424
 - std::__debug::map, 1430
 - std::__debug::multimap, 1436
 - std::__debug::multiset, 1442
-

- std::__debug::set, [1449](#)
- std::__debug::unordered_map, [1455](#)
- std::__debug::unordered_multimap, [1460](#)
- std::__debug::unordered_multiset, [1466](#)
- std::__debug::unordered_set, [1471](#)
- std::__debug::vector, [1478](#)
- _M_detach_single
 - __gnu_debug::_Safe_iterator, [1084](#)
 - __gnu_debug::_Safe_iterator_base, [1094](#)
 - __gnu_debug::_Safe_sequence, [1098](#)
 - __gnu_debug::_Safe_sequence_base, [1104](#)
 - __gnu_debug::basic_string, [1114](#)
- std::__debug::deque, [1411](#)
- std::__debug::forward_list, [1418](#)
- std::__debug::list, [1424](#)
- std::__debug::map, [1430](#)
- std::__debug::multimap, [1436](#)
- std::__debug::multiset, [1442](#)
- std::__debug::set, [1449](#)
- std::__debug::unordered_map, [1455](#)
- std::__debug::unordered_multimap, [1460](#)
- std::__debug::unordered_multiset, [1466](#)
- std::__debug::unordered_set, [1472](#)
- std::__debug::vector, [1479](#)
- _M_detach_singular
 - __gnu_debug::_Safe_sequence, [1099](#)
 - __gnu_debug::_Safe_sequence_base, [1104](#)
 - __gnu_debug::basic_string, [1114](#)
- std::__debug::deque, [1412](#)
- std::__debug::forward_list, [1418](#)
- std::__debug::list, [1425](#)
- std::__debug::map, [1431](#)
- std::__debug::multimap, [1437](#)
- std::__debug::multiset, [1443](#)
- std::__debug::set, [1449](#)
- std::__debug::unordered_map, [1455](#)
- std::__debug::unordered_multimap, [1461](#)
- std::__debug::unordered_multiset, [1466](#)
- std::__debug::unordered_set, [1472](#)
- std::__debug::vector, [1479](#)
- _M_dist
 - __gnu_parallel::_DRandomShufflingGlobalData, [1209](#)
- _M_elements_leftover
 - __gnu_parallel::_QSBThreadLocal, [1269](#)
- _M_end
 - __gnu_parallel::_Piece, [1260](#)
- _M_ext_buf
 - __gnu_cxx::enc_filebuf, [951](#)
 - __gnu_cxx::stdio_filebuf, [1027](#)
 - std::basic_filebuf, [1644](#)
- _M_ext_buf_size
 - __gnu_cxx::enc_filebuf, [951](#)
 - __gnu_cxx::stdio_filebuf, [1027](#)
 - std::basic_filebuf, [1644](#)
- _M_ext_next
 - __gnu_cxx::enc_filebuf, [951](#)
 - __gnu_cxx::stdio_filebuf, [1028](#)
 - std::basic_filebuf, [1644](#)
- _M_fill_initialize
 - std::deque, [2565](#)
- _M_finish_iterator
 - __gnu_parallel::__accumulate_selector, [1163](#)
 - __gnu_parallel::__adjacent_difference_selector, [1165](#)
 - __gnu_parallel::__count_if_selector, [1172](#)
 - __gnu_parallel::__count_selector, [1174](#)
 - __gnu_parallel::__fill_selector, [1176](#)
 - __gnu_parallel::__for_each_selector, [1181](#)
 - __gnu_parallel::__generate_selector, [1183](#)
 - __gnu_parallel::__generic_for_each_selector, [1186](#)

-
- __gnu_parallel::__identity_selector, 1187
 - __gnu_parallel::__inner_product_selector, 1190
 - __gnu_parallel::__replace_if_selector, 1199
 - __gnu_parallel::__replace_selector, 1202
 - __gnu_parallel::__transform1_selector, 1204
 - __gnu_parallel::__transform2_selector, 1205
 - _M_first
 - __gnu_parallel::__Job, 1223
 - _M_first_insert
 - __gnu_parallel::__LoserTree, 1233
 - __gnu_parallel::__LoserTree< _Tp, _Compare >, 1237
 - __gnu_parallel::__LoserTreeBase, 1241
 - _M_gcount
 - std::basic_fstream, 1715
 - std::basic_ifstream, 1774
 - std::basic_iostream, 1881
 - std::basic_istream, 1938
 - std::basic_istreamstream, 1999
 - std::basic_stringstream, 2354
 - _M_get
 - __gnu_cxx::free_list, 960
 - _M_get_distance
 - __gnu_debug::__Safe_iterator, 1085
 - _M_get_mutex
 - __gnu_debug::__Safe_iterator, 1085
 - __gnu_debug::__Safe_iterator_base, 1094
 - __gnu_debug::__Safe_sequence, 1099
 - __gnu_debug::__Safe_sequence_base, 1104
 - __gnu_debug::__basic_string, 1115
 - std::__debug::deque, 1412
 - std::__debug::forward_list, 1418
 - std::__debug::list, 1425
 - std::__debug::map, 1431
 - std::__debug::multimap, 1437
 - std::__debug::multiset, 1443
 - std::__debug::set, 1449
 - std::__debug::unordered_map, 1455
 - std::__debug::unordered_multimap, 1461
 - std::__debug::unordered_multiset, 1466
 - std::__debug::unordered_set, 1472
 - std::__debug::vector, 1479
 - _M_get_result
 - std::__basic_future, 1384
 - std::future, 2667
 - std::future< _Res & >, 2670
 - std::future< void >, 2673
 - std::shared_future, 3252
 - std::shared_future< _Res & >, 3255
 - std::shared_future< void >, 3258
 - _M_getloc
 - std::basic_fstream, 1666
 - std::basic_ifstream, 1736
 - std::basic_ios, 1793
 - std::basic_iostream, 1834
 - std::basic_istream, 1902
 - std::basic_istreamstream, 1962
 - std::basic_ofstream, 2021
 - std::basic_ostream, 2069
 - std::basic_ostreamstream, 2119
 - std::basic_stringstream, 2306
 - std::ios_base, 2745
 - _M_global
 - __gnu_parallel::__QSBThreadLocal, 1269
 - _M_in_beg
 - __gnu_cxx::enc_filebuf, 951
 - __gnu_cxx::stdio_filebuf, 1028
 - __gnu_cxx::stdio_sync_filebuf, 1057
 - std::basic_filebuf, 1644
 - std::basic_streambuf, 2193
 - std::basic_stringbuf, 2286
 - _M_in_cur
 - __gnu_cxx::enc_filebuf, 951
 - __gnu_cxx::stdio_filebuf, 1028
 - __gnu_cxx::stdio_sync_filebuf, 1058
 - std::basic_filebuf, 1645
 - std::basic_streambuf, 2194
-

-
- std::basic_stringbuf, 2286
 - _M_in_end
 - __gnu_cxx::enc_filebuf, 952
 - __gnu_cxx::stdio_filebuf, 1029
 - __gnu_cxx::stdio_sync_filebuf, 1058
 - std::basic_filebuf, 1645
 - std::basic_streambuf, 2194
 - std::basic_stringbuf, 2287
 - _M_incrementable
 - __gnu_debug::_Safe_iterator, 1085
 - _M_initial
 - __gnu_parallel::_QSBThreadLocal, 1269
 - _M_initialize_map
 - std::_Deque_base, 1529
 - std::deque, 2566
 - _M_insert
 - __gnu_cxx::free_list, 960
 - _M_invalidate
 - __gnu_debug::_Safe_iterator, 1085
 - __gnu_debug::_Safe_iterator_base, 1095
 - _M_invalidate_all
 - __gnu_debug::_Safe_sequence, 1099
 - __gnu_debug::_Safe_sequence_base, 1104
 - __gnu_debug::basic_string, 1115
 - std::__debug::deque, 1412
 - std::__debug::forward_list, 1418
 - std::__debug::list, 1425
 - std::__debug::map, 1431
 - std::__debug::multimap, 1437
 - std::__debug::multiset, 1443
 - std::__debug::set, 1450
 - std::__debug::unordered_map, 1455
 - std::__debug::unordered_multimap, 1461
 - std::__debug::unordered_multiset, 1466
 - std::__debug::unordered_set, 1472
 - std::__debug::vector, 1479
 - _M_invalidate_if
 - __gnu_debug::_Safe_sequence, 1099
 - __gnu_debug::basic_string, 1115
 - std::__debug::deque, 1412
 - std::__debug::forward_list, 1418
 - std::__debug::list, 1425
 - std::__debug::map, 1431
 - std::__debug::multimap, 1437
 - std::__debug::multiset, 1443
 - std::__debug::set, 1450
 - std::__debug::unordered_map, 1455
 - std::__debug::unordered_multimap, 1461
 - std::__debug::unordered_multiset, 1466
 - std::__debug::unordered_set, 1472
 - std::__debug::vector, 1479
 - __gnu_debug::basic_string, 1115
 - std::__debug::deque, 1412
 - std::__debug::forward_list, 1418
 - std::__debug::list, 1425
 - std::__debug::map, 1431
 - std::__debug::multimap, 1437
 - std::__debug::multiset, 1443
 - std::__debug::set, 1450
 - std::__debug::unordered_map, 1455
 - std::__debug::unordered_multimap, 1461
 - std::__debug::unordered_multiset, 1466
 - std::__debug::unordered_set, 1472
 - std::__debug::vector, 1479
 - _M_is_before_begin
 - __gnu_debug::_Safe_iterator, 1086
 - _M_is_begin
 - __gnu_debug::_Safe_iterator, 1086
 - _M_is_end
 - __gnu_debug::_Safe_iterator, 1086
 - _M_iterators
 - __gnu_debug::_Safe_sequence, 1100
 - __gnu_debug::_Safe_sequence_base, 1105
 - __gnu_debug::basic_string, 1115
 - std::__debug::deque, 1413
 - std::__debug::forward_list, 1419
 - std::__debug::list, 1426
 - std::__debug::map, 1432
 - std::__debug::multimap, 1438
 - std::__debug::multiset, 1444
 - std::__debug::set, 1451
 - std::__debug::unordered_map, 1457
 - std::__debug::unordered_multimap, 1462
 - std::__debug::unordered_multiset, 1468
 - std::__debug::unordered_set, 1473
 - std::__debug::vector, 1480
 - _M_key
 - __gnu_parallel::_LoserTreeBase::_Loser, 1242
 - _M_last
 - __gnu_parallel::_Job, 1224
-

-
- `_M_leftover_parts`
 - `__gnu_parallel::_QSBThreadLocal`, 1269
 - `_M_load`
 - `__gnu_parallel::_Job`, 1224
 - `_M_log_k`
 - `__gnu_parallel::_LoserTree`, 1233
 - `__gnu_parallel::_LoserTree< false, _Tp, _Compare >`, 1237
 - `__gnu_parallel::_LoserTreeBase`, 1241
 - `_M_losers`
 - `__gnu_parallel::_LoserTree`, 1233
 - `__gnu_parallel::_LoserTree< false, _Tp, _Compare >`, 1237
 - `__gnu_parallel::_LoserTreeBase`, 1241
 - `_M_mode`
 - `__gnu_cxx::enc_filebuf`, 952
 - `__gnu_cxx::stdio_filebuf`, 1029
 - `std::basic_filebuf`, 1646
 - `std::basic_stringbuf`, 2287
 - `_M_new_elements_at_back`
 - `std::deque`, 2566
 - `_M_new_elements_at_front`
 - `std::deque`, 2566
 - `_M_next`
 - `__gnu_debug::_Safe_iterator`, 1090
 - `__gnu_debug::_Safe_iterator_base`, 1095
 - `_M_num_bins`
 - `__gnu_parallel::_DRandomShufflingGlobalData`, 1209
 - `_M_num_bits`
 - `__gnu_parallel::_DRandomShufflingGlobalData`, 1209
 - `_M_num_threads`
 - `__gnu_parallel::_DRSSorterPU`, 1211
 - `__gnu_parallel::_PMWMSSortingData`, 1263
 - `__gnu_parallel::_QSBThreadLocal`, 1270
 - `_M_offsets`
 - `__gnu_parallel::_PMWMSSortingData`, 1263
 - `_M_out_beg`
 - `__gnu_cxx::enc_filebuf`, 952
 - `__gnu_cxx::stdio_filebuf`, 1030
 - `__gnu_cxx::stdio_sync_filebuf`, 1059
 - `std::basic_filebuf`, 1646
 - `std::basic_streambuf`, 2195
 - `std::basic_stringbuf`, 2287
 - `_M_out_cur`
 - `__gnu_cxx::enc_filebuf`, 953
 - `__gnu_cxx::stdio_filebuf`, 1030
 - `__gnu_cxx::stdio_sync_filebuf`, 1059
 - `std::basic_filebuf`, 1647
 - `std::basic_streambuf`, 2195
 - `std::basic_stringbuf`, 2288
 - `_M_out_end`
 - `__gnu_cxx::enc_filebuf`, 953
 - `__gnu_cxx::stdio_filebuf`, 1031
 - `__gnu_cxx::stdio_sync_filebuf`, 1060
 - `std::basic_filebuf`, 1647
 - `std::basic_streambuf`, 2196
 - `std::basic_stringbuf`, 2288
 - `_M_pback`
 - `__gnu_cxx::enc_filebuf`, 954
 - `__gnu_cxx::stdio_filebuf`, 1031
 - `std::basic_filebuf`, 1648
 - `_M_pback_cur_save`
 - `__gnu_cxx::enc_filebuf`, 954
 - `__gnu_cxx::stdio_filebuf`, 1031
 - `std::basic_filebuf`, 1648
 - `_M_pback_end_save`
 - `__gnu_cxx::enc_filebuf`, 954
 - `__gnu_cxx::stdio_filebuf`, 1032
 - `std::basic_filebuf`, 1648
 - `_M_pback_init`
 - `__gnu_cxx::enc_filebuf`, 954
 - `__gnu_cxx::stdio_filebuf`, 1032
 - `std::basic_filebuf`, 1648
 - `_M_pieces`
 - `__gnu_parallel::_PMWMSSortingData`, 1263
 - `_M_pop_back_aux`
-

-
- std::deque, 2567
 - _M_pop_front_aux
 - std::deque, 2567
 - _M_prior
 - __gnu_debug::_Safe_iterator, 1090
 - __gnu_debug::_Safe_iterator_base, 1096
 - _M_push_back_aux
 - std::deque, 2567
 - _M_push_front_aux
 - std::deque, 2567
 - _M_range_check
 - std::deque, 2568
 - std::vector, 3378
 - _M_range_initialize
 - std::deque, 2568
 - _M_reading
 - __gnu_cxx::enc_filebuf, 955
 - __gnu_cxx::stdio_filebuf, 1032
 - std::basic_filebuf, 1649
 - _M_reallocate_map
 - std::deque, 2569
 - _M_reserve_elements_at_back
 - std::deque, 2569
 - _M_reserve_elements_at_front
 - std::deque, 2570
 - _M_reserve_map_at_back
 - std::deque, 2570
 - _M_reserve_map_at_front
 - std::deque, 2570
 - _M_reset
 - __gnu_debug::_Safe_iterator, 1086
 - __gnu_debug::_Safe_iterator_base, 1095
 - _M_revalidate_singular
 - __gnu_debug::_Safe_sequence, 1099
 - __gnu_debug::_Safe_sequence_base, 1104
 - __gnu_debug::basic_string, 1115
 - std::__debug::deque, 1412
 - std::__debug::forward_list, 1418
 - std::__debug::list, 1425
 - std::__debug::map, 1431
 - std::__debug::multimap, 1437
 - std::__debug::multiset, 1443
 - std::__debug::set, 1450
 - std::__debug::unordered_map, 1456
 - std::__debug::unordered_multimap, 1461
 - std::__debug::unordered_multiset, 1467
 - std::__debug::unordered_set, 1473
 - std::__debug::vector, 1479
 - _M_samples
 - __gnu_parallel::_-PMWMSortingData, 1264
 - _M_sd
 - __gnu_parallel::_DRSSorterPU, 1212
 - _M_seed
 - __gnu_parallel::_DRSSorterPU, 1212
 - _M_sequence
 - __gnu_debug::_Safe_iterator, 1091
 - __gnu_debug::_Safe_iterator_base, 1096
 - _M_sequential_algorithm
 - __gnu_parallel::_adjacent_find_selector, 1166
 - __gnu_parallel::_find_first_of_selector, 1177
 - __gnu_parallel::_find_if_selector, 1179
 - __gnu_parallel::_mismatch_selector, 1193
 - _M_set_buffer
 - __gnu_cxx::enc_filebuf, 933
 - __gnu_cxx::stdio_filebuf, 1003
 - std::basic_filebuf, 1621
 - _M_set_node
 - std::_Deque_iterator, 1532
 - _M_singular
 - __gnu_debug::_Safe_iterator, 1087
 - __gnu_debug::_Safe_iterator_base, 1095
 - _M_source
 - __gnu_parallel::_-DRandomShufflingGlobalData, 1210
 - __gnu_parallel::_LoserTreeBase::_-Loser, 1242
-

-
- __gnu_parallel::_-
PMWMSSortingData, 1264
 - _M_starts
 - __gnu_parallel::_-
DRandomShufflingGlobalData,
1210
 - __gnu_parallel::_-
PMWMSSortingData, 1264
 - _M_sup
 - __gnu_parallel::LoserTreeBase::_-
Loser, 1243
 - _M_swap
 - __gnu_debug::_Safe_sequence,
1100
 - __gnu_debug::_Safe_sequence_-
base, 1104
 - __gnu_debug::basic_string, 1115
 - std::__debug::deque, 1413
 - std::__debug::forward_list, 1419
 - std::__debug::list, 1426
 - std::__debug::map, 1432
 - std::__debug::multimap, 1438
 - std::__debug::multiset, 1444
 - std::__debug::set, 1450
 - std::__debug::unordered_map, 1456
 - std::__debug::unordered_multimap,
1462
 - std::__debug::unordered_multiset,
1467
 - std::__debug::unordered_set, 1473
 - std::__debug::vector, 1480
 - _M_temporaries
 - __gnu_parallel::_-
DRandomShufflingGlobalData,
1210
 - _M_temporary
 - __gnu_parallel::_-
PMWMSSortingData, 1264
 - _M_transfer_from_if
 - __gnu_debug::_Safe_sequence,
1100
 - __gnu_debug::basic_string, 1116
 - std::__debug::deque, 1413
 - std::__debug::forward_list, 1419
 - std::__debug::list, 1426
 - std::__debug::map, 1432
 - std::__debug::multimap, 1438
 - std::__debug::multiset, 1444
 - std::__debug::set, 1450
 - std::__debug::unordered_map, 1456
 - std::__debug::unordered_multimap,
1462
 - std::__debug::unordered_multiset,
1467
 - std::__debug::unordered_set, 1473
 - std::__debug::vector, 1480
 - _M_unlink
 - __gnu_debug::_Safe_iterator, 1087
 - __gnu_debug::_Safe_iterator_base,
1095
 - _M_use_pointer
 - __gnu_parallel::LoserTreeTraits,
1252
 - _M_version
 - __gnu_debug::_Safe_iterator, 1091
 - __gnu_debug::_Safe_iterator_base,
1096
 - __gnu_debug::_Safe_sequence,
1101
 - __gnu_debug::_Safe_sequence_-
base, 1105
 - __gnu_debug::basic_string, 1161
 - std::__debug::deque, 1413
 - std::__debug::forward_list, 1420
 - std::__debug::list, 1426
 - std::__debug::map, 1433
 - std::__debug::multimap, 1439
 - std::__debug::multiset, 1445
 - std::__debug::set, 1451
 - std::__debug::unordered_map, 1457
 - std::__debug::unordered_multimap,
1463
 - std::__debug::unordered_multiset,
1468
 - std::__debug::unordered_set, 1474
 - std::__debug::vector, 1481
 - _M_w
 - std::_Base_bitset, 1524
 - _M_write
 - std::basic_fstream, 1666
 - std::basic_iostream, 1834
 - std::basic_ofstream, 2021
-

-
- std::basic_ostream, 2070
 - std::basic_ostringstream, 2120
 - std::basic_stringstream, 2306
 - _MultiwayMergeAlgorithm
 - __gnu_parallel, 430
 - _Parallelism
 - __gnu_parallel, 430
 - _PartialSumAlgorithm
 - __gnu_parallel, 431
 - _Piece
 - __gnu_parallel::__QSBThreadLocal, 1268
 - _PseudoSequence
 - __gnu_parallel::_PseudoSequence, 1266
 - _QSBThreadLocal
 - __gnu_parallel::_QSBThreadLocal, 1268
 - _RandomNumber
 - __gnu_parallel::_RandomNumber, 1271
 - _RestrictedBoundedConcurrentQueue
 - __gnu_parallel::_RestrictedBoundedConcurrentQueue, 1273
 - _Safe_iterator
 - __gnu_debug::_Safe_iterator, 1081, 1082
 - _Safe_iterator_base
 - __gnu_debug::_Safe_iterator_base, 1093
 - _SequenceIndex
 - __gnu_parallel, 430
 - _SortAlgorithm
 - __gnu_parallel, 431
 - _SplittingAlgorithm
 - __gnu_parallel, 431
 - _Temporary_buffer
 - std::_Temporary_buffer, 1564
 - _ThreadIndex
 - __gnu_parallel, 430
 - _Unchecked_flip
 - SGIextensions, 16
 - _Unchecked_reset
 - SGIextensions, 16
 - _Unchecked_set
 - SGIextensions, 16
 - _Unchecked_test
 - SGIextensions, 16
 - __base
 - __gnu_debug, 409
 - __begin1_iterator
 - __gnu_parallel::__inner_product_selector, 1190
 - __begin2_iterator
 - __gnu_parallel::__inner_product_selector, 1190
 - __bins_end
 - __gnu_parallel::_DRSSorterPU, 1211
 - __bit_allocate
 - __gnu_cxx::__detail, 401
 - __bit_free
 - __gnu_cxx::__detail, 401
 - __calc_borders
 - __gnu_parallel, 432
 - __check_dereferenceable
 - __gnu_debug, 409, 410
 - __check_singular
 - __gnu_debug, 410
 - __check_singular_aux
 - __gnu_debug, 410
 - __check_string
 - __gnu_debug, 411
 - __compare_and_swap
 - __gnu_parallel, 432
 - __compare_and_swap_32
 - __gnu_parallel, 432
 - __compare_and_swap_64
 - __gnu_parallel, 433
 - __ctype_type
 - std::basic_fstream, 1658
 - std::basic_ifstream, 1730
 - std::basic_ios, 1787
 - std::basic_iostream, 1826
 - std::basic_istream, 1896
 - std::basic_istream, 1956
 - std::basic_istream, 2014
 - std::basic_ofstream, 2064
 - std::basic_ostream, 2113
 - std::basic_ostringstream, 2298
 - std::basic_stringstream, 2298
 - __cxxabiv1::__forced_unwind, 816
-

-
- __decode2
 - __gnu_parallel, 433
 - __delete_min_insert
 - __gnu_parallel::_LoserTree, 1232
 - __gnu_parallel::_LoserTree< false, _Tp, _Compare >, 1235
 - __determine_samples
 - __gnu_parallel, 434
 - __encode2
 - __gnu_parallel, 434
 - __env_t
 - __gnu_profile, 488
 - __fetch_and_add
 - __gnu_parallel, 435
 - __fetch_and_add_32
 - __gnu_parallel, 435
 - __fetch_and_add_64
 - __gnu_parallel, 436
 - __final_insertion_sort
 - std, 641
 - __find
 - std, 642
 - __find_if
 - std, 642
 - __find_if_not
 - std, 643
 - __find_template
 - __gnu_parallel, 436–438
 - __for_each_template_random_access
 - __gnu_parallel, 439
 - __for_each_template_random_access_ed
 - __gnu_parallel, 439
 - __for_each_template_random_access_-omp_loop
 - __gnu_parallel, 440
 - __for_each_template_random_access_-omp_loop_static
 - __gnu_parallel, 441
 - __for_each_template_random_access_-workstealing
 - __gnu_parallel, 442
 - __gcd
 - std, 643
 - __genrand_bits
 - __gnu_parallel::_RandomNumber, 1271
 - __get__global_lock
 - __gnu_profile, 489
 - __get_min_source
 - __gnu_parallel::_LoserTree, 1232
 - __gnu_parallel::_LoserTree< false, _Tp, _Compare >, 1235
 - __gnu_parallel::_LoserTreeBase, 1240
 - __get_num_threads
 - __gnu_parallel::balanced_-quicksort_tag, 1289
 - __gnu_parallel::balanced_tag, 1290
 - __gnu_parallel::default_parallel_-tag, 1293
 - __gnu_parallel::exact_tag, 1295
 - __gnu_parallel::multiway_-mergesort_exact_tag, 1298
 - __gnu_parallel::multiway_-mergesort_sampling_tag, 1300
 - __gnu_parallel::multiway_-mergesort_tag, 1301
 - __gnu_parallel::omp_loop_static_-tag, 1303
 - __gnu_parallel::omp_loop_tag, 1304
 - __gnu_parallel::parallel_tag, 1307
 - __gnu_parallel::quicksort_tag, 1309
 - __gnu_parallel::sampling_tag, 1310
 - __gnu_parallel::unbalanced_tag, 1312
 - __glibcxx_check_erase
 - macros.h, 3608
 - __glibcxx_check_erase_after
 - macros.h, 3608
 - __glibcxx_check_erase_range
 - macros.h, 3608
 - __glibcxx_check_erase_range_after
 - macros.h, 3608
 - __glibcxx_check_heap_pred
 - macros.h, 3608
 - __glibcxx_check_insert
 - macros.h, 3608
 - __glibcxx_check_insert_after
 - macros.h, 3609
 - __glibcxx_check_insert_range
-

-
- macros.h, 3609
 - __glibcxx_check_insert_range_after
 - macros.h, 3609
 - __glibcxx_check_partitioned_lower
 - macros.h, 3610
 - __glibcxx_check_partitioned_lower_pred
 - macros.h, 3610
 - __glibcxx_check_partitioned_upper_pred
 - macros.h, 3610
 - __glibcxx_check_sorted_pred
 - macros.h, 3610
 - gnu_cxx, 364
 - _Bit_scan_forward, 388
 - __static_pointer_cast, 387
 - operator<, 392, 393
 - operator<=, 393, 394
 - operator>, 396, 397
 - operator>=, 398, 399
 - operator+, 389–391
 - operator==, 394–396
 - swap, 399
 - __gnu_cxx::_Caster, 904
 - __gnu_cxx::_Char_types, 904
 - __gnu_cxx::_ExtPtr_allocator, 905
 - __gnu_cxx::_Invalid_type, 907
 - __gnu_cxx::_Pointer_adapter, 907
 - __gnu_cxx::_Relative_pointer_impl, 910
 - __gnu_cxx::_Relative_pointer_impl<
 - const_Tp >, 910
 - __gnu_cxx::_Std_pointer_impl, 911
 - __gnu_cxx::_Unqualified_type, 912
 - __gnu_cxx::__common_pool_policy, 817
 - __gnu_cxx::__detail, 400
 - __bit_allocate, 401
 - __bit_free, 401
 - __num_bitmaps, 401
 - __num_blocks, 401
 - __gnu_cxx::__detail::_Bitmap_counter,
 - 818
 - __gnu_cxx::__detail::_Ffit_finder, 820
 - argument_type, 821
 - result_type, 821
 - __gnu_cxx::__detail::_mini_vector, 817
 - __gnu_cxx::__mt_alloc, 821
 - __gnu_cxx::__mt_alloc_base, 823
 - __gnu_cxx::__per_type_pool_policy, 824
 - __gnu_cxx::__pool< false >, 825
 - __gnu_cxx::__pool< true >, 826
 - __gnu_cxx::__pool_alloc, 827
 - __gnu_cxx::__pool_alloc_base, 829
 - __gnu_cxx::__pool_base, 831
 - __gnu_cxx::__rc_string_base, 832
 - __gnu_cxx::__scoped_lock, 835
 - __gnu_cxx::__versa_string, 835
 - ~__versa_string, 844
 - __versa_string, 840–844
 - append, 845–847
 - assign, 848–852
 - at, 852, 853
 - back, 853
 - begin, 854
 - c_str, 854
 - capacity, 854
 - cbegin, 855
 - cend, 855
 - clear, 855
 - compare, 855–859
 - copy, 860
 - crbegin, 860
 - crend, 861
 - data, 861
 - empty, 861
 - end, 861, 862
 - erase, 862, 863
 - find, 864, 865
 - find_first_not_of, 866–868
 - find_first_of, 868–870
 - find_last_not_of, 871–873
 - find_last_of, 873–875
 - front, 876
 - get_allocator, 876
 - insert, 876–881
 - length, 882
 - max_size, 882
 - npos, 903
 - operator+=, 883, 884
 - operator=, 884–886
 - push_back, 887
 - rbegin, 888
 - rend, 888, 889
 - replace, 889–896
 - reserve, 897
-

-
- resize, 898
 - rfind, 899, 900
 - shrink_to_fit, 901
 - size, 901
 - substr, 902
 - swap, 903
 - __gnu_cxx::annotate_base, 912
 - __gnu_cxx::array_allocator, 913
 - __gnu_cxx::array_allocator_base, 915
 - __gnu_cxx::binary_compose, 916
 - argument_type, 917
 - result_type, 917
 - __gnu_cxx::bitmap_allocator, 918
 - _M_allocate_single_object, 919
 - _M_deallocate_single_object, 920
 - __gnu_cxx::char_traits, 920
 - __gnu_cxx::character, 922
 - __gnu_cxx::condition_base, 923
 - __gnu_cxx::constant_binary_fun, 924
 - __gnu_cxx::constant_unary_fun, 924
 - __gnu_cxx::constant_void_fun, 925
 - __gnu_cxx::debug_allocator, 926
 - __gnu_cxx::enc_filebuf, 927
 - _M_buf, 950
 - _M_buf_locale, 950
 - _M_buf_size, 950
 - _M_create_pback, 932
 - _M_destroy_pback, 932
 - _M_ext_buf, 951
 - _M_ext_buf_size, 951
 - _M_ext_next, 951
 - _M_in_beg, 951
 - _M_in_cur, 951
 - _M_in_end, 952
 - _M_mode, 952
 - _M_out_beg, 952
 - _M_out_cur, 953
 - _M_out_end, 953
 - _M_pback, 954
 - _M_pback_cur_save, 954
 - _M_pback_end_save, 954
 - _M_pback_init, 954
 - _M_reading, 955
 - _M_set_buffer, 933
 - __streambuf_type, 931
 - char_type, 931
 - close, 933
 - eback, 933
 - egptr, 934
 - epptr, 934
 - gbump, 934
 - getloc, 935
 - gpptr, 935
 - imbue, 935
 - in_avail, 936
 - int_type, 931
 - is_open, 936
 - off_type, 931
 - open, 936, 937
 - overflow, 937
 - pbackfail, 938
 - pbase, 939
 - pbump, 939
 - pos_type, 932
 - pptr, 939
 - pubimbue, 940
 - pubseekoff, 940
 - pubseekpos, 941
 - pubsetbuf, 941
 - pubsync, 941
 - sbumpc, 942
 - seekoff, 942
 - seekpos, 942
 - setbuf, 943
 - setg, 943
 - setp, 944
 - sgetc, 944
 - sgetn, 944
 - showmanyc, 945
 - snextc, 945
 - sputbackc, 946
 - sputc, 946
 - sputn, 947
 - sungetc, 947
 - sync, 947
 - traits_type, 932
 - uflow, 948
 - underflow, 948
 - xsgetn, 949
 - xspn, 949
 - __gnu_cxx::encoding_char_traits, 955
 - __gnu_cxx::encoding_state, 957
-

-
- __gnu_cxx::forced_error, 958
 - what, 959
 - __gnu_cxx::free_list, 959
 - _M_clear, 960
 - _M_get, 960
 - _M_insert, 960
 - __gnu_cxx::hash_map, 961
 - __gnu_cxx::hash_multimap, 963
 - __gnu_cxx::hash_multiset, 965
 - __gnu_cxx::hash_set, 967
 - __gnu_cxx::limit_condition, 969
 - __gnu_cxx::limit_condition::always_ -
 adjustor, 970
 - __gnu_cxx::limit_condition::limit_ -
 adjustor, 970
 - __gnu_cxx::limit_condition::never_ -
 adjustor, 971
 - __gnu_cxx::malloc_allocator, 971
 - __gnu_cxx::new_allocator, 973
 - __gnu_cxx::project1st, 974
 - first_argument_type, 975
 - result_type, 975
 - second_argument_type, 975
 - __gnu_cxx::project2nd, 976
 - first_argument_type, 976
 - result_type, 976
 - second_argument_type, 977
 - __gnu_cxx::random_condition, 977
 - __gnu_cxx::random_condition::always_ -
 adjustor, 978
 - __gnu_cxx::random_condition::group_ -
 adjustor, 978
 - __gnu_cxx::random_condition::never_ -
 adjustor, 979
 - __gnu_cxx::rb_tree, 979
 - __gnu_cxx::recursive_init_error, 983
 - what, 983
 - __gnu_cxx::rope, 984
 - __gnu_cxx::select1st, 990
 - argument_type, 991
 - result_type, 991
 - __gnu_cxx::select2nd, 991
 - argument_type, 992
 - result_type, 992
 - __gnu_cxx::slist, 992
 - __gnu_cxx::stdio_filebuf, 995
 - ~stdio_filebuf, 1002
 - _M_buf, 1026
 - _M_buf_locale, 1027
 - _M_buf_size, 1027
 - _M_create_pback, 1003
 - _M_destroy_pback, 1003
 - _M_ext_buf, 1027
 - _M_ext_buf_size, 1027
 - _M_ext_next, 1028
 - _M_in_beg, 1028
 - _M_in_cur, 1028
 - _M_in_end, 1029
 - _M_mode, 1029
 - _M_out_beg, 1030
 - _M_out_cur, 1030
 - _M_out_end, 1031
 - _M_pback, 1031
 - _M_pback_cur_save, 1031
 - _M_pback_end_save, 1032
 - _M_pback_init, 1032
 - _M_reading, 1032
 - _M_set_buffer, 1003
 - __streambuf_type, 1000
 - char_type, 1000
 - close, 1004
 - eback, 1004
 - egptr, 1005
 - epptr, 1005
 - fd, 1006
 - file, 1006
 - gbump, 1006
 - getloc, 1007
 - gptr, 1007
 - imbue, 1008
 - in_avail, 1009
 - int_type, 1000
 - is_open, 1009
 - off_type, 1001
 - open, 1009, 1010
 - overflow, 1011
 - pbackfail, 1012
 - pbase, 1012
 - pbump, 1013
 - pos_type, 1001
 - pptr, 1013
 - pubimbue, 1014
-

- pubseekoff, 1014
- pubseekpos, 1015
- pubsetbuf, 1015
- pubsync, 1016
- sbumpc, 1016
- seekoff, 1016
- seekpos, 1017
- setbuf, 1017
- setg, 1018
- setp, 1019
- sgetc, 1019
- sgetn, 1019
- showmanyc, 1020
- snextc, 1021
- sputbackc, 1021
- sputc, 1021
- sputn, 1022
- stdio_filebuf, 1001, 1002
- sungetc, 1022
- sync, 1023
- traits_type, 1001
- uflow, 1023
- underflow, 1024
- xsgetn, 1025
- xspn, 1026
- __gnu_cxx::stdio_sync_filebuf, 1033
- _M_buf_locale, 1057
- _M_in_beg, 1057
- _M_in_cur, 1058
- _M_in_end, 1058
- _M_out_beg, 1059
- _M_out_cur, 1059
- _M_out_end, 1060
- __streambuf_type, 1037
- char_type, 1037
- eback, 1038
- egptr, 1039
- eptr, 1039
- file, 1040
- gbump, 1040
- getloc, 1041
- gptr, 1041
- imbue, 1042
- in_avail, 1042
- int_type, 1037
- off_type, 1037
- overflow, 1043
- pbackfail, 1043
- pbase, 1044
- pbump, 1045
- pos_type, 1038
- pptr, 1045
- pubimbue, 1046
- pubseekoff, 1046
- pubseekpos, 1046
- pubsetbuf, 1047
- pubsync, 1047
- sbumpc, 1048
- seekoff, 1048
- seekpos, 1048
- setbuf, 1049
- setg, 1049
- setp, 1050
- sgetc, 1050
- sgetn, 1051
- showmanyc, 1051
- snextc, 1052
- sputbackc, 1052
- sputc, 1053
- sputn, 1053
- sungetc, 1054
- sync, 1054
- traits_type, 1038
- uflow, 1055
- underflow, 1055
- xsgetn, 1056
- xspn, 1056
- __gnu_cxx::subtractive_rng, 1061
- argument_type, 1062
- operator(), 1062
- result_type, 1062
- subtractive_rng, 1062
- __gnu_cxx::temporary_buffer, 1063
- ~temporary_buffer, 1064
- begin, 1065
- end, 1065
- requested_size, 1065
- size, 1065
- temporary_buffer, 1064
- __gnu_cxx::throw_allocator_base, 1066
- __gnu_cxx::throw_allocator_limit, 1067

-
- __gnu_cxx::throw_allocator_random, 1069
 - __gnu_cxx::throw_value_base, 1071
 - __gnu_cxx::throw_value_limit, 1072
 - __gnu_cxx::throw_value_random, 1073
 - __gnu_cxx::typelist, 402
 - apply_generator, 402
 - __gnu_cxx::unary_compose, 1075
 - argument_type, 1076
 - result_type, 1076
 - __gnu_debug, 403
 - base, 409
 - check_dereferenceable, 409, 410
 - check_singular, 410
 - check_singular_aux, 410
 - check_string, 411
 - valid_range, 411
 - valid_range_aux, 411, 412
 - valid_range_aux2, 412
 - __gnu_debug::After_nth_from, 1076
 - __gnu_debug::BeforeBeginHelper, 1077
 - __gnu_debug::Equal_to, 1077
 - __gnu_debug::Not_equal_to, 1078
 - __gnu_debug::Safe_iterator, 1078
 - _M_attach, 1083
 - _M_attach_single, 1083
 - _M_attached_to, 1083
 - _M_before_dereferenceable, 1083
 - _M_can_compare, 1084
 - _M_dereferenceable, 1084
 - _M_detach, 1084
 - _M_detach_single, 1084
 - _M_get_distance, 1085
 - _M_get_mutex, 1085
 - _M_incrementable, 1085
 - _M_invalidate, 1085
 - _M_is_before_begin, 1086
 - _M_is_begin, 1086
 - _M_is_end, 1086
 - _M_next, 1090
 - _M_prior, 1090
 - _M_reset, 1086
 - _M_sequence, 1091
 - _M_singular, 1087
 - _M_unlink, 1087
 - _M_version, 1091
 - __Safe_iterator, 1081, 1082
 - base, 1087
 - operator_iterator, 1087
 - operator*, 1088
 - operator++, 1088
 - operator->, 1089
 - operator--, 1089
 - operator=, 1090
 - __gnu_debug::Safe_iterator_base, 1091
 - _M_attach, 1094
 - _M_attach_single, 1094
 - _M_attached_to, 1094
 - _M_can_compare, 1094
 - _M_detach, 1094
 - _M_detach_single, 1094
 - _M_get_mutex, 1094
 - _M_invalidate, 1095
 - _M_next, 1095
 - _M_prior, 1096
 - _M_reset, 1095
 - _M_sequence, 1096
 - _M_singular, 1095
 - _M_unlink, 1095
 - _M_version, 1096
 - _Safe_iterator_base, 1093
 - __gnu_debug::Safe_sequence, 1096
 - _M_attach, 1098
 - _M_attach_single, 1098
 - _M_const_iterators, 1100
 - _M_detach, 1098
 - _M_detach_all, 1098
 - _M_detach_single, 1098
 - _M_detach_singular, 1099
 - _M_get_mutex, 1099
 - _M_invalidate_all, 1099
 - _M_invalidate_if, 1099
 - _M_iterators, 1100
 - _M_revalidate_singular, 1099
 - _M_swap, 1100
 - _M_transfer_from_if, 1100
 - _M_version, 1101
 - __gnu_debug::Safe_sequence_base, 1101
 - ~_Safe_sequence_base, 1103
 - _M_attach, 1103
 - _M_attach_single, 1103
-

- [_M_const_iterators](#), 1105
- [_M_detach](#), 1103
- [_M_detach_all](#), 1103
- [_M_detach_single](#), 1104
- [_M_detach_singular](#), 1104
- [_M_get_mutex](#), 1104
- [_M_invalidate_all](#), 1104
- [_M_iterators](#), 1105
- [_M_revalidate_singular](#), 1104
- [_M_swap](#), 1104
- [_M_version](#), 1105
- [__gnu_debug::basic_string](#), 1106
 - [_M_attach](#), 1114
 - [_M_attach_single](#), 1114
 - [_M_const_iterators](#), 1160
 - [_M_detach](#), 1114
 - [_M_detach_all](#), 1114
 - [_M_detach_single](#), 1114
 - [_M_detach_singular](#), 1114
 - [_M_get_mutex](#), 1115
 - [_M_invalidate_all](#), 1115
 - [_M_invalidate_if](#), 1115
 - [_M_iterators](#), 1161
 - [_M_revalidate_singular](#), 1115
 - [_M_swap](#), 1115
 - [_M_transfer_from_if](#), 1116
 - [_M_version](#), 1161
- [append](#), 1116–1118
- [assign](#), 1119–1122
- [at](#), 1122, 1123
- [back](#), 1123
- [begin](#), 1123, 1124
- [c_str](#), 1124
- [capacity](#), 1124
- [cbegin](#), 1124
- [cend](#), 1124
- [clear](#), 1125
- [compare](#), 1125–1127
- [copy](#), 1128
- [crbegin](#), 1129
- [crend](#), 1129
- [data](#), 1129
- [empty](#), 1129
- [end](#), 1129
- [erase](#), 1130, 1131
- [find](#), 1131, 1132
- [find_first_not_of](#), 1133, 1134
- [find_first_of](#), 1135, 1136
- [find_last_not_of](#), 1136–1138
- [find_last_of](#), 1138–1140
- [front](#), 1140
- [get_allocator](#), 1140
- [insert](#), 1141–1145
- [length](#), 1146
- [max_size](#), 1146
- [npos](#), 1161
- [operator+=](#), 1146, 1147
- [push_back](#), 1148
- [rbegin](#), 1149
- [rend](#), 1149
- [replace](#), 1149–1155
- [reserve](#), 1156
- [resize](#), 1157
- [rfind](#), 1157–1159
- [shrink_to_fit](#), 1159
- [size](#), 1159
- [substr](#), 1159
- [swap](#), 1160
- [__gnu_internal](#), 413
- [__gnu_parallel](#), 413
 - [_AlgorithmStrategy](#), 430
 - [_BinIndex](#), 429
 - [_CASable](#), 429
 - [_CASable_bits](#), 480
 - [_CASable_mask](#), 480
 - [_FindAlgorithm](#), 430
 - [_MultiwayMergeAlgorithm](#), 430
 - [_Parallelism](#), 430
 - [_PartialSumAlgorithm](#), 431
 - [_SequenceIndex](#), 430
 - [_SortAlgorithm](#), 431
 - [_SplittingAlgorithm](#), 431
 - [_ThreadIndex](#), 430
 - [__calc_borders](#), 432
 - [__compare_and_swap](#), 432
 - [__compare_and_swap_32](#), 432
 - [__compare_and_swap_64](#), 433
 - [__decode2](#), 433
 - [__determine_samples](#), 434
 - [__encode2](#), 434
 - [__fetch_and_add](#), 435
 - [__fetch_and_add_32](#), 435

-
- `__fetch_and_add_64`, 436
 - `__find_template`, 436–438
 - `__for_each_template_random_-`
 - `access`, 439
 - `__for_each_template_random_-`
 - `access_ed`, 439
 - `__for_each_template_random_-`
 - `access_omp_loop`, 440
 - `__for_each_template_random_-`
 - `access_omp_loop_static`, 441
 - `__for_each_template_random_-`
 - `access_workstealing`, 442
 - `__is_sorted`, 442
 - `__median_of_three_iterators`, 443
 - `__merge_advance`, 443
 - `__merge_advance_movc`, 444
 - `__merge_advance_usual`, 445
 - `__parallel_merge_advance`, 445, 446
 - `__parallel_nth_element`, 447
 - `__parallel_partial_sort`, 447
 - `__parallel_partial_sum`, 448
 - `__parallel_partial_sum_basecase`, 448
 - `__parallel_partial_sum_linear`, 449
 - `__parallel_partition`, 450
 - `__parallel_random_shuffle`, 450
 - `__parallel_random_shuffle_drs`, 451
 - `__parallel_random_shuffle_drs_pu`, 451
 - `__parallel_sort`, 452–456
 - `__parallel_sort_qs`, 457
 - `__parallel_sort_qs_conquer`, 457
 - `__parallel_sort_qs_divide`, 458
 - `__parallel_sort_qsb`, 458
 - `__parallel_unique_copy`, 459
 - `__qsb_conquer`, 460
 - `__qsb_divide`, 460
 - `__qsb_local_sort_with_helping`, 461
 - `__random_number_pow2`, 461
 - `__rd_log2`, 462
 - `__round_up_to_pow2`, 462
 - `__search_template`, 463
 - `__sequential_multiway_merge`, 463
 - `__sequential_random_shuffle`, 464
 - `__shrink`, 464
 - `__shrink_and_double`, 465
 - `__yield`, 465
 - `equally_split`, 466
 - `equally_split_point`, 466
 - `list_partition`, 467
 - `max`, 467
 - `min`, 468
 - `multiseq_partition`, 468
 - `multiseq_selection`, 469
 - `multiway_merge`, 469
 - `multiway_merge_3_variant`, 471
 - `multiway_merge_4_variant`, 472
 - `multiway_merge_exact_splitting`, 473
 - `multiway_merge_loser_tree`, 473
 - `multiway_merge_loser_tree_-`
 - `sentinel`, 474
 - `multiway_merge_loser_tree_-`
 - `unguarded`, 475
 - `multiway_merge_sampling_-`
 - `splitting`, 475
 - `multiway_merge_sentinels`, 476
 - `parallel_multiway_merge`, 478
 - `parallel_sort_mwms`, 479
 - `parallel_sort_mwms_pu`, 479
 - `__gnu_parallel::_DRSSorterPU`, 1210
 - `__M_bins_begin`, 1211
 - `__M_num_threads`, 1211
 - `__M_sd`, 1212
 - `__M_seed`, 1212
 - `__bins_end`, 1211
 - `__gnu_parallel::_-`
 - `DRandomShufflingGlobalData`, 1207
 - `_DRandomShufflingGlobalData`, 1208
 - `_M_bin_proc`, 1209
 - `_M_dist`, 1209
 - `_M_num_bins`, 1209
 - `_M_num_bits`, 1209
 - `_M_source`, 1210
 - `_M_starts`, 1210
 - `_M_temporaries`, 1210
 - `__gnu_parallel::_DummyReduct`, 1212
 - `__gnu_parallel::_EqualFromLess`, 1213
 - `first_argument_type`, 1214
 - `result_type`, 1214
-

- second_argument_type, 1214
- __gnu_parallel::_EqualTo, 1215
 - first_argument_type, 1216
 - result_type, 1216
 - second_argument_type, 1216
- __gnu_parallel::_GuardedIterator, 1216
 - _GuardedIterator, 1217
 - operator_RAIter, 1218
 - operator<, 1219
 - operator<=, 1219
 - operator*, 1218
 - operator++, 1218
- __gnu_parallel::_IteratorPair, 1220
 - first, 1221
 - second, 1221
 - second_type, 1221
- __gnu_parallel::_IteratorTriple, 1222
- __gnu_parallel::_Job, 1223
 - _M_first, 1223
 - _M_last, 1224
 - _M_load, 1224
- __gnu_parallel::_Less, 1224
 - first_argument_type, 1226
 - result_type, 1226
 - second_argument_type, 1226
- __gnu_parallel::_Lexicographic, 1226
 - first_argument_type, 1228
 - result_type, 1228
 - second_argument_type, 1228
- __gnu_parallel::_LexicographicReverse, 1228
 - first_argument_type, 1230
 - result_type, 1230
 - second_argument_type, 1230
- __gnu_parallel::_LoserTree, 1230
 - _M_comp, 1233
 - _M_first_insert, 1233
 - _M_log_k, 1233
 - _M_losers, 1233
 - __delete_min_insert, 1232
 - __get_min_source, 1232
 - __insert_start, 1232
- __gnu_parallel::_LoserTree< false, _Tp, _Compare >, 1234
 - _M_comp, 1237
 - _M_first_insert, 1237
 - _M_log_k, 1237
 - _M_losers, 1237
 - __delete_min_insert, 1235
 - __get_min_source, 1235
 - __init_winner, 1236
 - __insert_start, 1236
- __gnu_parallel::_LoserTreeBase, 1238
 - ~_LoserTreeBase, 1239
 - _LoserTreeBase, 1239
 - _M_comp, 1241
 - _M_first_insert, 1241
 - _M_log_k, 1241
 - _M_losers, 1241
 - __get_min_source, 1240
 - __insert_start, 1240
- __gnu_parallel::_LoserTreeBase::_Loser, 1242
 - _M_key, 1242
 - _M_source, 1242
 - _M_sup, 1243
- __gnu_parallel::_LoserTreePointer, 1243
- __gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >, 1245
- __gnu_parallel::_LoserTreePointerBase, 1246
- __gnu_parallel::_-LoserTreePointerBase::_Loser, 1247
- __gnu_parallel::_-LoserTreePointerUnguarded, 1248
- __gnu_parallel::_-LoserTreePointerUnguarded< false, _Tp, _Compare >, 1249
- __gnu_parallel::_-LoserTreePointerUnguardedBase, 1250
- __gnu_parallel::_LoserTreeTraits, 1251
 - _M_use_pointer, 1252
- __gnu_parallel::_LoserTreeUnguarded, 1252
- __gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >, 1254
- __gnu_parallel::_-LoserTreeUnguardedBase, 1255

-
- __gnu_parallel::_Multiplies, 1256
 - first_argument_type, 1258
 - result_type, 1258
 - second_argument_type, 1258
 - __gnu_parallel::_Nothing, 1258
 - operator(), 1259
 - __gnu_parallel::_PMWMSSortingData, 1262
 - _M_num_threads, 1263
 - _M_offsets, 1263
 - _M_pieces, 1263
 - _M_samples, 1264
 - _M_source, 1264
 - _M_starts, 1264
 - _M_temporary, 1264
 - __gnu_parallel::_Piece, 1259
 - _M_begin, 1260
 - _M_end, 1260
 - __gnu_parallel::_Plus, 1260
 - first_argument_type, 1262
 - result_type, 1262
 - second_argument_type, 1262
 - __gnu_parallel::_PseudoSequence, 1265
 - _PseudoSequence, 1266
 - begin, 1266
 - end, 1266
 - __gnu_parallel::_-
 - PseudoSequenceIterator, 1267
 - __gnu_parallel::_QSBThreadLocal, 1267
 - _M_elements_leftover, 1269
 - _M_global, 1269
 - _M_initial, 1269
 - _M_leftover_parts, 1269
 - _M_num_threads, 1270
 - _Piece, 1268
 - _QSBThreadLocal, 1268
 - __gnu_parallel::_RandomNumber, 1270
 - _RandomNumber, 1271
 - _genrand_bits, 1271
 - operator(), 1271, 1272
 - __gnu_parallel::_-
 - RestrictedBoundedConcurrentQueue, 1272
 - ~_RestrictedBoundedConcurrentQueue, 1273
 - _RestrictedBoundedConcurrentQueue, 1273
 - pop_back, 1274
 - pop_front, 1274
 - push_front, 1274
 - __gnu_parallel::_SamplingSorter, 1275
 - __gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering >, 1275
 - __gnu_parallel::_Settings, 1276
 - accumulate_minimal_n, 1278
 - adjacent_difference_minimal_n, 1278
 - cache_line_size, 1278
 - count_minimal_n, 1278
 - fill_minimal_n, 1279
 - find_increasing_factor, 1279
 - find_initial_block_size, 1279
 - find_maximum_block_size, 1279
 - find_scale_factor, 1279
 - find_sequential_search_size, 1280
 - for_each_minimal_n, 1280
 - generate_minimal_n, 1280
 - get, 1277
 - L1_cache_size, 1280
 - L2_cache_size, 1280
 - max_element_minimal_n, 1280
 - merge_minimal_n, 1281
 - merge_oversampling, 1281
 - min_element_minimal_n, 1281
 - multiway_merge_minimal_k, 1281
 - multiway_merge_minimal_n, 1281
 - multiway_merge_oversampling, 1282
 - nth_element_minimal_n, 1282
 - partial_sort_minimal_n, 1282
 - partial_sum_dilation, 1282
 - partial_sum_minimal_n, 1282
 - partition_chunk_share, 1283
 - partition_chunk_size, 1283
 - partition_minimal_n, 1283
 - qsb_steals, 1283
 - random_shuffle_minimal_n, 1283
 - replace_minimal_n, 1284
 - search_minimal_n, 1284
 - set, 1278
-

- set_difference_minimal_n, 1284
- set_intersection_minimal_n, 1284
- set_symmetric_difference_-
 - minimal_n, 1284
- set_union_minimal_n, 1285
- sort_minimal_n, 1285
- sort_mwms_oversampling, 1285
- sort_qs_num_samples_preset, 1285
- sort_qsb_base_case_maximal_n, 1285
- TLB_size, 1285
- transform_minimal_n, 1286
- unique_copy_minimal_n, 1286
- __gnu_parallel::__SplitConsistently, 1286
- __gnu_parallel::__SplitConsistently<
 - false, _RAIter, _Compare, _SortingPlacesIterator >, 1287
- __gnu_parallel::__SplitConsistently<
 - true, _RAIter, _Compare, _SortingPlacesIterator >, 1287
- __gnu_parallel::__accumulate_binop_-
 - reduct, 1161
- __gnu_parallel::__accumulate_selector, 1162
 - _M_finish_iterator, 1163
 - operator(), 1163
- __gnu_parallel::__adjacent_difference_-
 - selector, 1164
 - _M_finish_iterator, 1165
- __gnu_parallel::__adjacent_find_-
 - selector, 1165
 - _M_sequential_algorithm, 1166
 - operator(), 1166
- __gnu_parallel::__binder1st, 1167
 - argument_type, 1168
 - result_type, 1168
- __gnu_parallel::__binder2nd, 1169
 - argument_type, 1170
 - result_type, 1170
- __gnu_parallel::__count_if_selector, 1170
 - _M_finish_iterator, 1172
 - operator(), 1171
- __gnu_parallel::__count_selector, 1172
 - _M_finish_iterator, 1174
 - operator(), 1173
- __gnu_parallel::__fill_selector, 1174
 - _M_finish_iterator, 1176
 - operator(), 1175
- __gnu_parallel::__find_first_of_selector, 1176
 - _M_sequential_algorithm, 1177
 - operator(), 1177
- __gnu_parallel::__find_if_selector, 1178
 - _M_sequential_algorithm, 1179
 - operator(), 1179
- __gnu_parallel::__for_each_selector, 1180
 - _M_finish_iterator, 1181
 - operator(), 1181
- __gnu_parallel::__generate_selector, 1181
 - _M_finish_iterator, 1183
 - operator(), 1182
- __gnu_parallel::__generic_find_selector, 1183
 - _M_finish_iterator, 1183
 - operator(), 1182
- __gnu_parallel::__generic_for_each_-
 - selector, 1184
 - _M_finish_iterator, 1186
- __gnu_parallel::__identity_selector, 1186
 - _M_finish_iterator, 1187
 - operator(), 1187
- __gnu_parallel::__inner_product_-
 - selector, 1188
 - _M_finish_iterator, 1190
- __begin1_iterator, 1190
- __begin2_iterator, 1190
- __inner_product_selector, 1189
 - operator(), 1189
- __gnu_parallel::__max_element_reduct, 1190
 - _M_finish_iterator, 1190
- __gnu_parallel::__min_element_reduct, 1191
 - _M_finish_iterator, 1190
- __gnu_parallel::__mismatch_selector, 1192
 - _M_sequential_algorithm, 1193
 - operator(), 1193
- __gnu_parallel::__multiway_merge_-
 - 3_variant_sentinel_switch, 1193
- __gnu_parallel::__multiway_merge_-
 - 3_variant_sentinel_switch<

- true, [_RAIterIterator](#), [_RAIter3](#), [_DifferenceTp](#), [_Compare >](#), [1194](#)
- [__gnu_parallel::__multiway_merge_4_variant_sentinel_switch](#), [1195](#)
- [__gnu_parallel::__multiway_merge_4_variant_sentinel_switch<true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >](#), [1195](#)
- [__gnu_parallel::__multiway_merge_k_variant_sentinel_switch](#), [1196](#)
- [__gnu_parallel::__multiway_merge_k_variant_sentinel_switch<false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >](#), [1197](#)
- [__gnu_parallel::__replace_if_selector](#), [1197](#)
 - [_M_finish_iterator](#), [1199](#)
 - [_new_val](#), [1199](#)
 - [_replace_if_selector](#), [1198](#)
 - [operator\(\)](#), [1199](#)
- [__gnu_parallel::__replace_selector](#), [1200](#)
 - [_M_finish_iterator](#), [1202](#)
 - [_new_val](#), [1202](#)
 - [_replace_selector](#), [1201](#)
 - [operator\(\)](#), [1201](#)
- [__gnu_parallel::__transform1_selector](#), [1202](#)
 - [_M_finish_iterator](#), [1204](#)
 - [operator\(\)](#), [1203](#)
- [__gnu_parallel::__transform2_selector](#), [1204](#)
 - [_M_finish_iterator](#), [1205](#)
 - [operator\(\)](#), [1205](#)
- [__gnu_parallel::__unary_negate](#), [1206](#)
 - [argument_type](#), [1207](#)
 - [result_type](#), [1207](#)
- [__gnu_parallel::balanced_quicksort_tag](#), [1288](#)
 - [__get_num_threads](#), [1289](#)
 - [set_num_threads](#), [1289](#)
- [__gnu_parallel::balanced_tag](#), [1289](#)
- [__get_num_threads](#), [1290](#)
- [set_num_threads](#), [1290](#)
- [__gnu_parallel::constant_size_blocks_tag](#), [1291](#)
- [__gnu_parallel::default_parallel_tag](#), [1292](#)
 - [__get_num_threads](#), [1293](#)
 - [set_num_threads](#), [1293](#)
- [__gnu_parallel::equal_split_tag](#), [1293](#)
- [__gnu_parallel::exact_tag](#), [1294](#)
 - [__get_num_threads](#), [1295](#)
 - [set_num_threads](#), [1295](#)
- [__gnu_parallel::find_tag](#), [1296](#)
- [__gnu_parallel::growing_blocks_tag](#), [1297](#)
 - [__gnu_parallel::multiway_mergesort_exact_tag](#), [1297](#)
 - [__get_num_threads](#), [1298](#)
 - [set_num_threads](#), [1298](#)
- [__gnu_parallel::multiway_mergesort_sampling_tag](#), [1299](#)
 - [__get_num_threads](#), [1300](#)
 - [set_num_threads](#), [1300](#)
- [__gnu_parallel::multiway_mergesort_tag](#), [1300](#)
 - [__get_num_threads](#), [1301](#)
 - [set_num_threads](#), [1301](#)
- [__gnu_parallel::omp_loop_static_tag](#), [1302](#)
 - [__get_num_threads](#), [1303](#)
 - [set_num_threads](#), [1303](#)
- [__gnu_parallel::omp_loop_tag](#), [1303](#)
 - [__get_num_threads](#), [1304](#)
 - [set_num_threads](#), [1304](#)
- [__gnu_parallel::parallel_tag](#), [1305](#)
 - [__get_num_threads](#), [1307](#)
 - [parallel_tag](#), [1307](#)
 - [set_num_threads](#), [1307](#)
- [__gnu_parallel::quicksort_tag](#), [1308](#)
 - [__get_num_threads](#), [1309](#)
 - [set_num_threads](#), [1309](#)
- [__gnu_parallel::sampling_tag](#), [1309](#)
 - [__get_num_threads](#), [1310](#)
 - [set_num_threads](#), [1310](#)
- [__gnu_parallel::sequential_tag](#), [1311](#)
- [__gnu_parallel::unbalanced_tag](#), [1311](#)

-
- __get_num_threads, [1312](#)
 - set_num_threads, [1312](#)
 - __gnu_pbds, [480](#)
 - __gnu_pbds::associative_container_tag, [1313](#)
 - __gnu_pbds::basic_hash_table, [1314](#)
 - __gnu_pbds::basic_hash_tag, [1315](#)
 - __gnu_pbds::basic_tree, [1316](#)
 - __gnu_pbds::basic_tree_tag, [1317](#)
 - __gnu_pbds::binary_heap_tag, [1318](#)
 - __gnu_pbds::binomial_heap_tag, [1318](#)
 - __gnu_pbds::cc_hash_table, [1319](#)
 - __gnu_pbds::cc_hash_tag, [1321](#)
 - __gnu_pbds::container_base, [1322](#)
 - __gnu_pbds::container_tag, [1324](#)
 - __gnu_pbds::container_traits, [1324](#)
 - __gnu_pbds::detail::value_type_base< Key, Mapped, Allocator, false >, [1325](#)
 - __gnu_pbds::detail::value_type_base< Key, Mapped, Allocator, true >, [1326](#)
 - __gnu_pbds::detail::value_type_base< Key, null_mapped_type, Allocator, false >, [1327](#)
 - __gnu_pbds::detail::value_type_base< Key, null_mapped_type, Allocator, true >, [1328](#)
 - __gnu_pbds::gp_hash_table, [1328](#)
 - __gnu_pbds::gp_hash_tag, [1331](#)
 - __gnu_pbds::list_update, [1332](#)
 - __gnu_pbds::list_update_tag, [1333](#)
 - __gnu_pbds::null_mapped_type, [1334](#)
 - __gnu_pbds::ov_tree_tag, [1335](#)
 - __gnu_pbds::pairing_heap_tag, [1336](#)
 - __gnu_pbds::pat_trie_tag, [1336](#)
 - __gnu_pbds::priority_queue_tag, [1337](#)
 - __gnu_pbds::rb_tree_tag, [1338](#)
 - __gnu_pbds::rc_binomial_heap_tag, [1339](#)
 - __gnu_pbds::sequence_tag, [1340](#)
 - __gnu_pbds::splay_tree_tag, [1341](#)
 - __gnu_pbds::string_tag, [1342](#)
 - __gnu_pbds::thin_heap_tag, [1343](#)
 - __gnu_pbds::tree, [1344](#)
 - __gnu_pbds::tree_tag, [1346](#)
 - __gnu_pbds::trie, [1347](#)
 - __gnu_pbds::trie_tag, [1348](#)
 - __gnu_profile, [483](#)
 - __env_t, [488](#)
 - __get__global_lock, [489](#)
 - __profcxx_init, [489](#)
 - __report, [489](#)
 - __gnu_profile::__container_size_info, [1349](#)
 - __gnu_profile::__container_size_stack_info, [1351](#)
 - __gnu_profile::__hashfunc_info, [1352](#)
 - __gnu_profile::__hashfunc_stack_info, [1353](#)
 - __gnu_profile::__list2vector_info, [1355](#)
 - __gnu_profile::__map2umap_info, [1356](#)
 - __gnu_profile::__map2umap_stack_info, [1357](#)
 - __gnu_profile::__object_info_base, [1359](#)
 - __gnu_profile::__reentrance_guard, [1360](#)
 - __gnu_profile::__stack_hash, [1360](#)
 - __gnu_profile::__stack_info_base, [1361](#)
 - __gnu_profile::__trace_base, [1361](#)
 - __gnu_profile::__trace_container_size, [1362](#)
 - __gnu_profile::__trace_hash_func, [1363](#)
 - __gnu_profile::__trace_hashtable_size, [1364](#)
 - __gnu_profile::__trace_map2umap, [1365](#)
 - __gnu_profile::__trace_vector_size, [1366](#)
 - __gnu_profile::__trace_vector_to_list, [1367](#)
 - __gnu_profile::__vector2list_info, [1368](#)
 - __gnu_profile::__vector2list_stack_info, [1370](#)
 - __gnu_profile::__warning_data, [1371](#)
 - __gnu_sequential, [489](#)
 - __heap_select
 - std, [643](#), [644](#)
 - __init_winner
 - __gnu_parallel::LoserTree< false, _Tp, _Compare >, [1236](#)
 - __inner_product_selector
 - __gnu_parallel::__inner_product_selector, [1189](#)
 - __inplace_stable_partition
-

-
- std, [644](#)
 - __inplace_stable_sort
 - std, [644](#), [645](#)
 - __insert_start
 - __gnu_parallel::_LoserTree, [1232](#)
 - __gnu_parallel::_LoserTree< false, _Tp, _Compare >, [1236](#)
 - __gnu_parallel::_LoserTreeBase, [1240](#)
 - __insertion_sort
 - std, [645](#)
 - __introsort_loop
 - std, [645](#), [646](#)
 - __invoke
 - std, [739](#)
 - __ioinit
 - std, [739](#)
 - __is_sorted
 - __gnu_parallel, [442](#)
 - __iterator_category
 - iterators, [288](#)
 - __lg
 - std, [646](#)
 - __match_flag
 - std::regex_constants, [801](#)
 - __median
 - SGIextensions, [14](#)
 - __median_of_three_iterators
 - __gnu_parallel, [443](#)
 - __merge_adaptive
 - std, [646](#), [647](#)
 - __merge_advance
 - __gnu_parallel, [443](#)
 - __merge_advance_movc
 - __gnu_parallel, [444](#)
 - __merge_advance_usual
 - __gnu_parallel, [445](#)
 - __merge_backward
 - std, [647](#)
 - __merge_without_buffer
 - std, [648](#)
 - __move_median_first
 - std, [648](#)
 - __new_val
 - __gnu_parallel::__replace_if_selector, [1199](#)
 - __gnu_parallel::__replace_selector, [1202](#)
 - __num_bitmaps
 - __gnu_cxx::__detail, [401](#)
 - __num_blocks
 - __gnu_cxx::__detail, [401](#)
 - __num_get_type
 - std::basic_fstream, [1658](#)
 - std::basic_ifstream, [1730](#)
 - std::basic_ios, [1787](#)
 - std::basic_iostream, [1826](#)
 - std::basic_istream, [1896](#)
 - std::basic_istreamstream, [1956](#)
 - std::basic_ofstream, [2015](#)
 - std::basic_ostream, [2064](#)
 - std::basic_ostreamstream, [2114](#)
 - std::basic_stringstream, [2298](#)
 - __num_put_type
 - std::basic_fstream, [1658](#)
 - std::basic_ifstream, [1730](#)
 - std::basic_ios, [1787](#)
 - std::basic_iostream, [1827](#)
 - std::basic_istream, [1897](#)
 - std::basic_istreamstream, [1956](#)
 - std::basic_ofstream, [2015](#)
 - std::basic_ostream, [2064](#)
 - std::basic_ostreamstream, [2114](#)
 - std::basic_stringstream, [2298](#)
 - __parallel_merge_advance
 - __gnu_parallel, [445](#), [446](#)
 - __parallel_nth_element
 - __gnu_parallel, [447](#)
 - __parallel_partial_sort
 - __gnu_parallel, [447](#)
 - __parallel_partial_sum
 - __gnu_parallel, [448](#)
 - __parallel_partial_sum_basecase
 - __gnu_parallel, [448](#)
 - __parallel_partial_sum_linear
 - __gnu_parallel, [449](#)
 - __parallel_partition
 - __gnu_parallel, [450](#)
 - __parallel_random_shuffle
 - __gnu_parallel, [450](#)
 - __parallel_random_shuffle_drs
 - __gnu_parallel, [451](#)
-

-
- __parallel_random_shuffle_drs_pu
 - __gnu_parallel, 451
 - __parallel_sort
 - __gnu_parallel, 452–456
 - __parallel_sort_qs
 - __gnu_parallel, 457
 - __parallel_sort_qs_conquer
 - __gnu_parallel, 457
 - __parallel_sort_qs_divide
 - __gnu_parallel, 458
 - __parallel_sort_qsb
 - __gnu_parallel, 458
 - __parallel_unique_copy
 - __gnu_parallel, 459
 - __partition
 - std, 649
 - __profcxx_init
 - __gnu_profile, 489
 - __qsb_conquer
 - __gnu_parallel, 460
 - __qsb_divide
 - __gnu_parallel, 460
 - __qsb_local_sort_with_helping
 - __gnu_parallel, 461
 - __random_number_pow2
 - __gnu_parallel, 461
 - __rd_log2
 - __gnu_parallel, 462
 - __replace_if_selector
 - __gnu_parallel::__replace_if_-selector, 1198
 - __replace_selector
 - __gnu_parallel::__replace_selector, 1201
 - __report
 - __gnu_profile, 489
 - __reverse
 - std, 649, 650
 - __rotate
 - std, 650
 - __rotate_adaptive
 - std, 651
 - __round_up_to_pow2
 - __gnu_parallel, 462
 - __search_n
 - std, 651, 652
 - __search_template
 - __gnu_parallel, 463
 - __sequential_multiway_merge
 - __gnu_parallel, 463
 - __sequential_random_shuffle
 - __gnu_parallel, 464
 - __shrink
 - __gnu_parallel, 464
 - __shrink_and_double
 - __gnu_parallel, 465
 - __stable_partition_adaptive
 - std, 652
 - __static_pointer_cast
 - __gnu_cxx, 387
 - __streambuf_type
 - __gnu_cxx::enc_filebuf, 931
 - __gnu_cxx::stdio_filebuf, 1000
 - __gnu_cxx::stdio_sync_filebuf, 1037
 - std::basic_filebuf, 1618
 - std::basic_streambuf, 2172
 - std::basic_stringbuf, 2263
 - __syntax_option
 - std::regex_constants, 802
 - __unguarded_insertion_sort
 - std, 652, 653
 - __unguarded_linear_insert
 - std, 653
 - __unguarded_partition
 - std, 653, 654
 - __unguarded_partition_pivot
 - std, 654
 - __unique_copy
 - std, 654–656
 - __valid_range
 - __gnu_debug, 411
 - __valid_range_aux
 - __gnu_debug, 411, 412
 - __valid_range_aux2
 - __gnu_debug, 412
 - __verbose_terminate_handler
 - exceptions, 34
 - __versa_string
 - __gnu_cxx::__versa_string, 840–844
 - __yield
-

-
- __gnu_parallel, 465
 - a
 - std::extreme_value_distribution, 2616
 - std::weibull_distribution, 3401
 - abi, 490
 - abs
 - complex_numbers, 42
 - accumulate
 - std, 657
 - accumulate_minimal_n
 - __gnu_parallel::_Settings, 1278
 - acos
 - complex_numbers, 42
 - std, 658
 - acosh
 - complex_numbers, 43
 - std, 658
 - Adaptors for pointers to functions, 274
 - Adaptors for pointers to members, 276
 - addressof
 - std, 658
 - adjacent_difference
 - std, 658, 659
 - adjacent_difference_minimal_n
 - __gnu_parallel::_Settings, 1278
 - adjacent_find
 - non_mutating_algorithms, 156
 - adjustfield
 - std::basic_fstream, 1715
 - std::basic_ifstream, 1775
 - std::basic_ios, 1810
 - std::basic_iostream, 1882
 - std::basic_istream, 1939
 - std::basic_istream, 2000
 - std::basic_ofstream, 2050
 - std::basic_ostream, 2097
 - std::basic_ostringstream, 2148
 - std::basic_stringstream, 2354
 - std::ios_base, 2752
 - advance
 - std, 660
 - algo.h, 3404
 - algbase.h, 3418
 - algorithm, 3420, 3422
 - algorithmfwd.h, 3422, 3429
 - Algorithms, 128
 - all
 - std::bitset, 2386
 - std::locale, 2868
 - all_of
 - non_mutating_algorithms, 156
 - allocate_shared
 - pointer_abstractions, 71
 - std::shared_ptr, 3266
 - allocator.h, 3442
 - allocator_type
 - std::set, 3231
 - Allocators, 207
 - alpha
 - std::gamma_distribution, 2677
 - any
 - std::bitset, 2386
 - any_of
 - non_mutating_algorithms, 157
 - app
 - std::basic_fstream, 1715
 - std::basic_ifstream, 1775
 - std::basic_ios, 1810
 - std::basic_iostream, 1882
 - std::basic_istream, 1939
 - std::basic_istream, 2000
 - std::basic_ofstream, 2050
 - std::basic_ostream, 2097
 - std::basic_ostringstream, 2148
 - std::basic_stringstream, 2354
 - std::ios_base, 2752
 - append
 - __gnu_cxx::__versa_string, 845–847
 - __gnu_debug::basic_string, 1116–1118
 - std::basic_string, 2206–2209
 - apply
 - numeric_arrays, 95
 - apply_generator
 - __gnu_cxx::typelist, 402
 - arg
 - complex_numbers, 43
 - std, 660
 - argument_type
-

-
- __gnu_cxx::__detail::_Ffit_finder, 821
 - __gnu_cxx::binary_compose, 917
 - __gnu_cxx::select1st, 991
 - __gnu_cxx::select2nd, 992
 - __gnu_cxx::subtractive_rng, 1062
 - __gnu_cxx::unary_compose, 1076
 - __gnu_parallel::__binder1st, 1168
 - __gnu_parallel::__binder2nd, 1170
 - __gnu_parallel::__unary_negate, 1207
 - std::Maybe_unary_or_binary_-function< _Res, _T1 >, 1549
 - std::binder1st, 2372
 - std::binder2nd, 2373
 - std::const_mem_fun_ref_t, 2465
 - std::const_mem_fun_t, 2467
 - std::hash< __gnu_cxx::throw_-value_limit >, 2706
 - std::hash< __gnu_cxx::throw_-value_random >, 2708
 - std::hash< __shared_ptr< _Tp, _Lp > >, 2711
 - std::hash< shared_ptr< _Tp > >, 2714
 - std::hash< unique_ptr< _Tp, _Dp > >, 2719
 - std::logical_not, 2880
 - std::mem_fun_ref_t, 2930
 - std::mem_fun_t, 2932
 - std::negate, 3027
 - std::pointer_to_unary_function, 3163
 - std::unary_function, 3339
 - std::unary_negate, 3341
 - Arithmetic Classes, 270
 - array, 3443
 - array_allocator.h, 3444
 - asin
 - complex_numbers, 43
 - std, 660
 - asinh
 - complex_numbers, 43
 - std, 661
 - assign
 - __gnu_cxx::__versa_string, 848–852
 - __gnu_debug::basic_string, 1119–1122
 - std::basic_regex, 2161–2164
 - std::basic_string, 2209–2213
 - std::deque, 2570, 2571
 - std::forward_list, 2632, 2633
 - std::list, 2840, 2841
 - std::vector, 3378, 3379
 - assoc_container.hpp, 3444
 - assoc_laguerre
 - tr1_math_spec_func, 120
 - assoc_legendre
 - tr1_math_spec_func, 120
 - Associative, 27
 - async
 - futures, 59
 - at
 - __gnu_cxx::__versa_string, 852, 853
 - __gnu_debug::basic_string, 1122, 1123
 - std::basic_string, 2213, 2214
 - std::deque, 2572
 - std::map, 2894
 - std::vector, 3379, 3380
 - atan
 - complex_numbers, 43
 - std, 661
 - atanh
 - complex_numbers, 44
 - std, 661
 - ate
 - std::basic_fstream, 1715
 - std::basic_ifstream, 1775
 - std::basic_ios, 1811
 - std::basic_iostream, 1882
 - std::basic_istream, 1939
 - std::basic_istreamstream, 2000
 - std::basic_ofstream, 2050
 - std::basic_ostream, 2097
 - std::basic_ostreamstream, 2148
 - std::basic_stringstream, 2354
 - std::ios_base, 2752
 - atomic, 3446
-

-
- atomic_0.h, [3452](#)
 - atomic_2.h, [3453](#)
 - atomic_base.h, [3454](#)
 - atomic_char
 - atomics, [218](#)
 - atomic_char16_t
 - atomics, [218](#)
 - atomic_char32_t
 - atomics, [218](#)
 - atomic_int
 - atomics, [218](#)
 - atomic_int_fast16_t
 - atomics, [218](#)
 - atomic_int_fast32_t
 - atomics, [218](#)
 - atomic_int_fast64_t
 - atomics, [219](#)
 - atomic_int_fast8_t
 - atomics, [219](#)
 - atomic_int_least16_t
 - atomics, [219](#)
 - atomic_int_least32_t
 - atomics, [219](#)
 - atomic_int_least64_t
 - atomics, [219](#)
 - atomic_int_least8_t
 - atomics, [219](#)
 - atomic_intmax_t
 - atomics, [220](#)
 - atomic_intptr_t
 - atomics, [220](#)
 - atomic_llong
 - atomics, [220](#)
 - atomic_long
 - atomics, [220](#)
 - atomic_ptrdiff_t
 - atomics, [220](#)
 - atomic_schar
 - atomics, [220](#)
 - atomic_short
 - atomics, [221](#)
 - atomic_size_t
 - atomics, [221](#)
 - atomic_uchar
 - atomics, [221](#)
 - atomic_uint
 - atomics, [221](#)
 - atomic_uint_fast16_t
 - atomics, [221](#)
 - atomic_uint_fast32_t
 - atomics, [221](#)
 - atomic_uint_fast64_t
 - atomics, [222](#)
 - atomic_uint_fast8_t
 - atomics, [222](#)
 - atomic_uint_least16_t
 - atomics, [222](#)
 - atomic_uint_least32_t
 - atomics, [222](#)
 - atomic_uint_least64_t
 - atomics, [222](#)
 - atomic_uint_least8_t
 - atomics, [222](#)
 - atomic_uintmax_t
 - atomics, [223](#)
 - atomic_uintptr_t
 - atomics, [223](#)
 - atomic_ullong
 - atomics, [223](#)
 - atomic_ulong
 - atomics, [223](#)
 - atomic_ushort
 - atomics, [223](#)
 - atomic_wchar_t
 - atomics, [223](#)
 - atomic_word.h, [3456](#)
 - atomicity.h, [3456](#)
 - Atomics, [208](#)
 - atomics
 - _GLIBCXX_ATOMIC_-
PROPERTY, [217](#)
 - atomic_char, [218](#)
 - atomic_char16_t, [218](#)
 - atomic_char32_t, [218](#)
 - atomic_int, [218](#)
 - atomic_int_fast16_t, [218](#)
 - atomic_int_fast32_t, [218](#)
 - atomic_int_fast64_t, [219](#)
 - atomic_int_fast8_t, [219](#)
 - atomic_int_least16_t, [219](#)
 - atomic_int_least32_t, [219](#)
 - atomic_int_least64_t, [219](#)
-

- atomic_int_least8_t, 219
- atomic_intmax_t, 220
- atomic_intptr_t, 220
- atomic_llong, 220
- atomic_long, 220
- atomic_ptrdiff_t, 220
- atomic_schar, 220
- atomic_short, 221
- atomic_size_t, 221
- atomic_uchar, 221
- atomic_uint, 221
- atomic_uint_fast16_t, 221
- atomic_uint_fast32_t, 221
- atomic_uint_fast64_t, 222
- atomic_uint_fast8_t, 222
- atomic_uint_least16_t, 222
- atomic_uint_least32_t, 222
- atomic_uint_least64_t, 222
- atomic_uint_least8_t, 222
- atomic_uintmax_t, 223
- atomic_uintptr_t, 223
- atomic_ullong, 223
- atomic_ulong, 223
- atomic_ushort, 223
- atomic_wchar_t, 223
- kill_dependency, 224
- memory_order, 224
- auto_ptr
 - std::auto_ptr, 1599, 1600
- auto_ptr.h, 3457
- awk
 - std::regex_constants, 804
- b
 - std::extreme_value_distribution, 2616
 - std::weibull_distribution, 3401
- back
 - __gnu_cxx::__versa_string, 853
 - __gnu_debug::basic_string, 1123
 - std::basic_string, 2214
 - std::deque, 2573
 - std::list, 2842
 - std::queue, 3177
 - std::vector, 3380
- back_insert_iterator
 - std::back_insert_iterator, 1606
- back_inserter
 - iterators, 289
- backward_warning.h, 3458
- bad
 - std::basic_fstream, 1667
 - std::basic_ifstream, 1737
 - std::basic_ios, 1794
 - std::basic_iostream, 1835
 - std::basic_istream, 1902
 - std::basic_istreamstream, 1962
 - std::basic_ofstream, 2021
 - std::basic_ostream, 2070
 - std::basic_ostringstream, 2120
 - std::basic_stringstream, 2306
- badbit
 - std::basic_fstream, 1716
 - std::basic_ifstream, 1775
 - std::basic_ios, 1811
 - std::basic_iostream, 1882
 - std::basic_istream, 1939
 - std::basic_istreamstream, 2000
 - std::basic_ofstream, 2050
 - std::basic_ostream, 2098
 - std::basic_ostringstream, 2148
 - std::basic_stringstream, 2355
 - std::ios_base, 2752
- balanced_quicksort.h, 3458
- base
 - __gnu_debug::_Safe_iterator, 1087
 - std::discard_block_engine, 2590
 - std::independent_bits_engine, 2724
 - std::reverse_iterator, 3221
 - std::shuffle_order_engine, 3270
- base.h, 3459, 3461
- basefield
 - std::basic_fstream, 1716
 - std::basic_ifstream, 1776
 - std::basic_ios, 1811
 - std::basic_iostream, 1883
 - std::basic_istream, 1940
 - std::basic_istreamstream, 2001
 - std::basic_ofstream, 2051
 - std::basic_ostream, 2098
 - std::basic_ostringstream, 2149
 - std::basic_stringstream, 2355

-
- std::ios_base, 2753
 - basic
 - std::regex_constants, 804
 - basic_file.h, 3461
 - basic_filebuf
 - std::basic_filebuf, 1620
 - basic_fstream
 - std::basic_fstream, 1664, 1665
 - basic_ifstream
 - std::basic_ifstream, 1735, 1736
 - basic_ios
 - std::basic_ios, 1793
 - basic_ios.h, 3461
 - basic_ios.tcc, 3462
 - basic_iostream
 - std::basic_iostream, 1833
 - basic_istream
 - std::basic_istream, 1901
 - basic_istreamstream
 - std::basic_istreamstream, 1961
 - basic_iterator.h, 3462
 - basic_ofstream
 - std::basic_ofstream, 2019, 2020
 - basic_ostream
 - std::basic_ostream, 2069
 - basic_ostreamstream
 - std::basic_ostreamstream, 2118, 2119
 - basic_regex
 - std::basic_regex, 2158–2160
 - basic_streambuf
 - std::basic_streambuf, 2174
 - basic_string
 - std::basic_string, 2202–2205
 - basic_string.h, 3463
 - basic_string.tcc, 3467
 - basic_stringbuf
 - std::basic_stringbuf, 2265
 - basic_stringstream
 - std::basic_stringstream, 2304, 2305
 - basic_types.hpp, 3468
 - before_begin
 - std::forward_list, 2633, 2634
 - beg
 - std::basic_fstream, 1716
 - std::basic_ifstream, 1776
 - std::basic_ios, 1812
 - std::basic_iostream, 1883
 - std::basic_istream, 1940
 - std::basic_istreamstream, 2001
 - std::basic_ofstream, 2051
 - std::basic_ostream, 2098
 - std::basic_ostreamstream, 2149
 - std::basic_stringstream, 2355
 - std::ios_base, 2753
 - begin
 - __gnu_cxx::__versa_string, 854
 - __gnu_cxx::temporary_buffer, 1065
 - __gnu_debug::basic_string, 1123, 1124
 - __gnu_parallel::_PseudoSequence, 1266
 - numeric_arrays, 95, 96
 - std, 661, 662
 - std::_Temporary_buffer, 1564
 - std::basic_string, 2215
 - std::deque, 2573
 - std::forward_list, 2634
 - std::list, 2842
 - std::map, 2894
 - std::match_results, 2918
 - std::multimap, 2990
 - std::multiset, 3011
 - std::set, 3237
 - std::vector, 3381
 - Bernoulli Distributions, 317
 - bernoulli_distribution
 - std::bernoulli_distribution, 2363
 - beta
 - std::gamma_distribution, 2677
 - tr1_math_spec_func, 120
 - binary
 - std::basic_fstream, 1717
 - std::basic_ifstream, 1776
 - std::basic_ios, 1812
 - std::basic_iostream, 1883
 - std::basic_istream, 1940
 - std::basic_istreamstream, 2001
 - std::basic_ofstream, 2051
 - std::basic_ostream, 2099
 - std::basic_ostreamstream, 2149
 - std::basic_stringstream, 2356
 - std::ios_base, 2753
-

- Binary Search, [201](#)
- binary_search
 - binary_search_algorithms, [202](#)
- binary_search_algorithms
 - binary_search, [202](#)
 - equal_range, [203](#), [204](#)
 - lower_bound, [204](#), [205](#)
 - upper_bound, [206](#)
- bind
 - binders, [127](#)
 - std, [662](#)
- bind1st
 - binders, [127](#)
- bind2nd
 - binders, [127](#)
- Binder Classes, [125](#)
- binders
 - bind, [127](#)
 - bind1st, [127](#)
 - bind2nd, [127](#)
- binders.h, [3468](#)
- bitmap_allocator.h, [3469](#)
- _BALLOC_ALIGN_BYTES, [3470](#)
- bitset, [3471–3473](#)
 - std::bitset, [2384](#), [2385](#)
- boolalpha
 - std, [663](#)
 - std::basic_fstream, [1717](#)
 - std::basic_ifstream, [1776](#)
 - std::basic_ios, [1812](#)
 - std::basic_istream, [1883](#)
 - std::basic_istream, [1940](#)
 - std::basic_istream, [2001](#)
 - std::basic_ofstream, [2052](#)
 - std::basic_ostream, [2099](#)
 - std::basic_ostringstream, [2149](#)
 - std::basic_stringstream, [2356](#)
 - std::ios_base, [2754](#)
- Boolean Operations Classes, [272](#)
- boost_concept_check.h, [3474](#)
- c
 - std::queue, [3179](#)
- c++0x_warning.h, [3475](#)
- c++allocator.h, [3475](#)
- c++config.h, [3475](#)
- c++io.h, [3480](#)
- c++locale.h, [3481](#)
- c++locale_internal.h, [3481](#)
- c_str
 - __gnu_cxx::__versa_string, [854](#)
 - __gnu_debug::basic_string, [1124](#)
 - std::basic_string, [2215](#)
- cache_line_size
 - __gnu_parallel::_Settings, [1278](#)
- call_once
 - mutexes, [74](#)
 - std::once_flag, [3126](#)
- capacity
 - __gnu_cxx::__versa_string, [854](#)
 - __gnu_debug::basic_string, [1124](#)
 - std::basic_string, [2215](#)
 - std::vector, [3381](#)
- cassert, [3481](#)
- cast.h, [3482](#)
- category
 - std::locale, [2862](#)
- cbefore_begin
 - std::forward_list, [2634](#)
- cbegin
 - __gnu_cxx::__versa_string, [855](#)
 - __gnu_debug::basic_string, [1124](#)
 - std::basic_string, [2216](#)
 - std::deque, [2574](#)
 - std::forward_list, [2635](#)
 - std::list, [2842](#)
 - std::map, [2895](#)
 - std::match_results, [2919](#)
 - std::multimap, [2991](#)
 - std::multiset, [3012](#)
 - std::set, [3237](#)
 - std::vector, [3382](#)
- ccomplex, [3483](#)
- cctype, [3483](#)
- cend
 - __gnu_cxx::__versa_string, [855](#)
 - __gnu_debug::basic_string, [1124](#)
 - std::basic_string, [2216](#)
 - std::deque, [2574](#)
 - std::forward_list, [2635](#)
 - std::list, [2843](#)
 - std::map, [2895](#)

- std::match_results, 2919
- std::multimap, 2991
- std::multiset, 3012
- std::set, 3237
- std::vector, 3382
- cerr
 - std, 739
- cerrno, 3484
- cfenv, 3484
- cfloat, 3484, 3485
- char_traits.h, 3485
- char_type
 - __gnu_cxx::enc_filebuf, 931
 - __gnu_cxx::stdio_filebuf, 1000
 - __gnu_cxx::stdio_sync_filebuf, 1037
- std::__ctype_abstract_base, 1393
- std::basic_filebuf, 1618
- std::basic_fstream, 1659
- std::basic_ifstream, 1731
- std::basic_ios, 1788
- std::basic_iostream, 1827
- std::basic_istream, 1897
- std::basic_istream, 1957
- std::basic_ofstream, 2015
- std::basic_ostream, 2064
- std::basic_ostringstream, 2114
- std::basic_streambuf, 2172
- std::basic_stringbuf, 2263
- std::basic_stringstream, 2299
- std::collate, 2444
- std::collate_byname, 2451
- std::ctype, 2470
- std::ctype< char >, 2487
- std::ctype< wchar_t >, 2503
- std::ctype_byname, 2520
- std::ctype_byname< char >, 2536
- std::istreambuf_iterator, 2813
- std::messages, 2935
- std::messages_byname, 2940
- std::money_get, 2948
- std::money_put, 2954
- std::moneypunct, 2961
- std::moneypunct_byname, 2974
- std::num_get, 3045
- std::num_put, 3066
- std::numpunct, 3113
- std::numpunct_byname, 3121
- std::ostream_iterator, 3128
- std::ostreambuf_iterator, 3133
- std::time_get, 3298
- std::time_get_byname, 3310
- std::time_put, 3321
- std::time_put_byname, 3326
- checkers.h, 3486
- chrono, 3486
- cin
 - std, 739
- cinttypes, 3489, 3490
- ciso646, 3490
- classic
 - std::locale, 2864
- classic_table
 - std::ctype< char >, 2488
 - std::ctype_byname< char >, 2537
- clear
 - __gnu_cxx::__versa_string, 855
 - __gnu_debug::basic_string, 1125
 - std::basic_fstream, 1667
 - std::basic_ifstream, 1737
 - std::basic_ios, 1794
 - std::basic_iostream, 1835
 - std::basic_istream, 1902
 - std::basic_istream, 1963
 - std::basic_ofstream, 2022
 - std::basic_ostream, 2071
 - std::basic_ostringstream, 2121
 - std::basic_string, 2216
 - std::basic_stringstream, 2307
 - std::deque, 2574
 - std::forward_list, 2635
 - std::list, 2843
 - std::map, 2895
 - std::multimap, 2991
 - std::multiset, 3012
 - std::set, 3238
 - std::vector, 3382
- climits, 3490
- locale, 3491
- clog
 - std, 739
- close

-
- __gnu_cxx::enc_filebuf, 933
 - __gnu_cxx::stdio_filebuf, 1004
 - std::basic_filebuf, 1621
 - std::basic_fstream, 1667
 - std::basic_ifstream, 1738
 - std::basic_ofstream, 2022
 - cmath, 3491, 3494
 - code
 - std::regex_error, 3196
 - codecv.h, 3497
 - codecv_specializations.h, 3498
 - collate
 - std::collate, 2444
 - std::locale, 2868
 - std::regex_constants, 804
 - combine
 - std::locale, 2864
 - comp_ellint_1
 - tr1_math_spec_func, 120
 - comp_ellint_2
 - tr1_math_spec_func, 120
 - comp_ellint_3
 - tr1_math_spec_func, 121
 - compare
 - __gnu_cxx::__versa_string, 855–859
 - __gnu_debug::basic_string, 1125–1127
 - std::basic_string, 2216–2220
 - std::collate, 2445
 - std::collate_byname, 2451
 - std::sub_match, 3288, 3289
 - Comparison Classes, 271
 - compatibility.h, 3499
 - compiletime_settings.h, 3500
 - _GLIBCXX_ASSERTIONS, 3500
 - _GLIBCXX_CALL, 3501
 - _GLIBCXX_RANDOM_-SHUFFLE_CONSIDER_L1, 3501
 - _GLIBCXX_RANDOM_-SHUFFLE_CONSIDER_TLB, 3501
 - _GLIBCXX_SCALE_DOWN_-FPU, 3502
 - _GLIBCXX_VERBOSE_LEVEL, 3502
 - complex, 3502, 3507
 - std::complex, 2456
 - Complex Numbers, 37
 - complex.h, 3509
 - complex_numbers
 - abs, 42
 - acos, 42
 - acosh, 43
 - arg, 43
 - asin, 43
 - asinh, 43
 - atan, 43
 - atanh, 44
 - conj, 44
 - cos, 44
 - cosh, 44
 - exp, 44
 - fabs, 45
 - log, 45
 - log10, 45
 - norm, 45
 - operator<<, 50
 - operator>>, 52
 - operator*, 46, 47
 - operator*=, 47
 - operator+, 47, 48
 - operator+=, 48
 - operator-, 48, 49
 - operator=, 49
 - operator/, 49, 50
 - operator/=: 50
 - operator=, 51
 - operator==, 51
 - polar, 52
 - pow, 52, 53
 - sin, 53
 - sinh, 53
 - sqrt, 54
 - tan, 54
 - tanh, 54
 - compose1
 - SGIextensions, 17
 - compose2
 - SGIextensions, 17
-

- concept_check.h, 3509
- concurrency.h, 3509
- Concurrency, 30
- cond_dealtor.hpp, 3510
- Condition Variables, 55
- condition_variable, 3510
- condition_variables
 - cv_status, 55
- conf_hyperg
 - tr1_math_spec_func, 121
- conj
 - complex_numbers, 44
- const_iterator
 - std::set, 3231
- const_pointer
 - std::set, 3231
- const_pointer_cast
 - std, 663
- const_reference
 - std::set, 3231
- const_reverse_iterator
 - std::set, 3231
- constant0
 - SGIextensions, 17
- constant1
 - SGIextensions, 17
- constant2
 - SGIextensions, 17
- constructors_destructor_fn_imps.hpp, 3511
- container_base_dispatch.hpp, 3512
- container_type
 - std::back_insert_iterator, 1605
 - std::front_insert_iterator, 2653
 - std::insert_iterator, 2734
- Containers, 24
- copy
 - __gnu_cxx::__versa_string, 860
 - __gnu_debug::basic_string, 1128
 - mutating_algorithms, 132
 - std::basic_string, 2220
- copy_backward
 - mutating_algorithms, 132
- copy_exception
 - exceptions, 34
- copy_if
 - mutating_algorithms, 133
- copy_n
 - mutating_algorithms, 133
 - SGIextensions, 18
- copyfmt
 - std::basic_fstream, 1668
 - std::basic_ifstream, 1738
 - std::basic_ios, 1795
 - std::basic_iostream, 1835
 - std::basic_istream, 1903
 - std::basic_istreamstream, 1963
 - std::basic_ofstream, 2023
 - std::basic_ostream, 2071
 - std::basic_ostreamstream, 2121
 - std::basic_stringstream, 2307
- cos
 - complex_numbers, 44
- cosh
 - complex_numbers, 44
- count
 - non_mutating_algorithms, 157
 - std::bitset, 2387
 - std::map, 2895
 - std::multimap, 2991
 - std::multiset, 3012
 - std::set, 3238
- count_if
 - non_mutating_algorithms, 158
- count_minimal_n
 - __gnu_parallel::_Settings, 1278
- cout
 - std, 739
- cpp_type_traits.h, 3512
- cpu_defines.h, 3512
- crbegin
 - __gnu_cxx::__versa_string, 860
 - __gnu_debug::basic_string, 1129
 - std::basic_string, 2221
 - std::deque, 2574
 - std::list, 2843
 - std::map, 2896
 - std::multimap, 2992
 - std::multiset, 3013
 - std::set, 3238
 - std::vector, 3382
- cref

-
- std, 663
 - cregex_token_iterator
 - regex, 235
 - crend
 - __gnu_cxx::__versa_string, 861
 - __gnu_debug::basic_string, 1129
 - std::basic_string, 2221
 - std::deque, 2575
 - std::list, 2843
 - std::map, 2896
 - std::multimap, 2992
 - std::multiset, 3013
 - std::set, 3239
 - std::vector, 3382
 - csetjmp, 3513
 - cshift
 - numeric_arrays, 96
 - csignal, 3513
 - cstdarg, 3513, 3514
 - cstdbool, 3514
 - cstddef, 3515
 - cstdint, 3515
 - cstdio, 3515, 3516
 - cstdlib, 3516, 3517
 - cstring, 3517
 - csub_match
 - regex, 235
 - ctgmath, 3517, 3518
 - ctime, 3518
 - ctype
 - std::ctype< char >, 2487
 - std::ctype< wchar_t >, 2503
 - std::locale, 2868
 - ctype_base.h, 3518
 - ctype_inline.h, 3519
 - ctype_noninline.h, 3519
 - cur
 - std::basic_fstream, 1717
 - std::basic_ifstream, 1777
 - std::basic_ios, 1812
 - std::basic_iostream, 1884
 - std::basic_istream, 1941
 - std::basic_istream, 2002
 - std::basic_ofstream, 2052
 - std::basic_ostream, 2099
 - std::basic_ostringstream, 2150
 - std::basic_stringstream, 2356
 - std::ios_base, 2754
 - curr_symbol
 - std::moneypunct, 2963
 - std::moneypunct_byname, 2975
 - current_exception
 - exceptions, 34
 - cv_status
 - condition_variables, 55
 - cwchar, 3519, 3520
 - cwctype, 3520, 3521
 - cxxabi.h, 3521
 - cxxabi_forced.h, 3523
 - cxxabi_tweaks.h, 3523
 - cyl_bessel_i
 - tr1_math_spec_func, 121
 - cyl_bessel_j
 - tr1_math_spec_func, 121
 - cyl_bessel_k
 - tr1_math_spec_func, 121
 - cyl_neumann
 - tr1_math_spec_func, 122
 - data
 - __gnu_cxx::__versa_string, 861
 - __gnu_debug::basic_string, 1129
 - std::basic_string, 2221
 - std::vector, 3383
 - date_order
 - std::time_get, 3299
 - std::time_get_byname, 3311
 - debug.h, 3524
 - debug_allocator.h, 3524
 - debug_map_base.hpp, 3525
 - dec
 - std, 663
 - std::basic_fstream, 1717
 - std::basic_ifstream, 1777
 - std::basic_ios, 1813
 - std::basic_iostream, 1884
 - std::basic_istream, 1941
 - std::basic_istream, 2002
 - std::basic_ofstream, 2052
 - std::basic_ostream, 2099
 - std::basic_ostringstream, 2150
 - std::basic_stringstream, 2356
-

-
- std::ios_base, 2754
 - decimal, 3525
 - Decimal Floating-Point Arithmetic, 125
 - decimal128
 - std::decimal::decimal128, 2550
 - decimal32
 - std::decimal::decimal32, 2552
 - decimal32_to_long_long
 - std::decimal, 798
 - decimal64
 - std::decimal::decimal64, 2554
 - decimal_point
 - std::moneypunct, 2963
 - std::moneypunct_byname, 2975
 - std::numpunct, 3114
 - std::numpunct_byname, 3121
 - denorm_absent
 - std, 641
 - denorm_indeterminate
 - std, 641
 - denorm_present
 - std, 641
 - denorm_min
 - std::numeric_limits, 3080
 - densities
 - std::piecewise_constant_distribution, 3147
 - std::piecewise_linear_distribution, 3153
 - deque, 3537, 3538
 - std::deque, 2562–2564
 - deque.tcc, 3539
 - Diagnostics, 29
 - difference_type
 - std::back_insert_iterator, 1605
 - std::front_insert_iterator, 2653
 - std::insert_iterator, 2734
 - std::istream_iterator, 2810
 - std::istreambuf_iterator, 2813
 - std::iterator, 2818
 - std::ostream_iterator, 3128
 - std::ostreambuf_iterator, 3133
 - std::raw_storage_iterator, 3191
 - std::reverse_iterator, 3219
 - std::set, 3232
 - digits
 - std::__numeric_limits_base, 1496
 - std::numeric_limits, 3082
 - digits10
 - std::__numeric_limits_base, 1496
 - std::numeric_limits, 3082
 - discard
 - std::discard_block_engine, 2590
 - std::independent_bits_engine, 2724
 - std::linear_congruential_engine, 2828
 - std::shuffle_order_engine, 3270
 - discard_block_engine
 - std::discard_block_engine, 2588–2590
 - distance
 - SGIextensions, 18
 - std, 664
 - do_compare
 - std::collate, 2445
 - std::collate_byname, 2451
 - do_curr_symbol
 - std::moneypunct, 2963
 - std::moneypunct_byname, 2975
 - do_date_order
 - std::time_get, 3300
 - std::time_get_byname, 3311
 - do_decimal_point
 - std::moneypunct, 2964
 - std::moneypunct_byname, 2976
 - std::numpunct, 3115
 - std::numpunct_byname, 3122
 - do_falsename
 - std::numpunct, 3115
 - std::numpunct_byname, 3122
 - do_frac_digits
 - std::moneypunct, 2964
 - std::moneypunct_byname, 2976
 - do_get
 - std::money_get, 2949
 - std::num_get, 3046–3053
 - do_get_date
 - std::time_get, 3300
 - std::time_get_byname, 3312
 - do_get_monthname
 - std::time_get, 3301
 - std::time_get_byname, 3312
-

-
- do_get_time
 - std::time_get, 3302
 - std::time_get_byname, 3313
 - do_get_weekday
 - std::time_get, 3302
 - std::time_get_byname, 3314
 - do_get_year
 - std::time_get, 3303
 - std::time_get_byname, 3314
 - do_grouping
 - std::moneypunct, 2965
 - std::moneypunct_byname, 2977
 - std::numpunct, 3115
 - std::numpunct_byname, 3122
 - do_hash
 - std::collate, 2446
 - std::collate_byname, 2452
 - do_is
 - std::__ctype_abstract_base, 1393
 - std::ctype, 2471
 - std::ctype< wchar_t >, 2504
 - std::ctype_byname, 2521
 - do_narrow
 - std::__ctype_abstract_base, 1394
 - std::ctype, 2472
 - std::ctype< char >, 2488
 - std::ctype< wchar_t >, 2504, 2505
 - std::ctype_byname, 2521, 2522
 - std::ctype_byname< char >, 2537
 - do_neg_format
 - std::moneypunct, 2965
 - std::moneypunct_byname, 2977
 - do_negative_sign
 - std::moneypunct, 2965
 - std::moneypunct_byname, 2977
 - do_out
 - std::__codecvt_abstract_base, 1386
 - std::codecvt, 2415
 - std::codecvt< _InternT, _ExternT, encoding_state >, 2421
 - std::codecvt< char, char, mbstate_t >, 2426
 - std::codecvt< wchar_t, char, mbstate_t >, 2431
 - std::codecvt_byname, 2438
 - do_pos_format
 - std::moneypunct, 2966
 - std::moneypunct_byname, 2978
 - do_positive_sign
 - std::moneypunct, 2966
 - std::moneypunct_byname, 2978
 - do_put
 - std::money_put, 2955
 - std::num_put, 3067–3070
 - std::time_put, 3322
 - std::time_put_byname, 3327
 - do_scan_is
 - std::__ctype_abstract_base, 1395
 - std::ctype, 2473
 - std::ctype< wchar_t >, 2506
 - std::ctype_byname, 2523
 - do_scan_not
 - std::__ctype_abstract_base, 1396
 - std::ctype, 2473
 - std::ctype< wchar_t >, 2506
 - std::ctype_byname, 2523
 - do_thousands_sep
 - std::moneypunct, 2967
 - std::moneypunct_byname, 2979
 - std::numpunct, 3116
 - std::numpunct_byname, 3123
 - do_tolower
 - std::__ctype_abstract_base, 1396, 1397
 - std::ctype, 2474
 - std::ctype< char >, 2489
 - std::ctype< wchar_t >, 2507
 - std::ctype_byname, 2524
 - std::ctype_byname< char >, 2538
 - do_toupper
 - std::__ctype_abstract_base, 1397, 1398
 - std::ctype, 2475
 - std::ctype< char >, 2490
 - std::ctype< wchar_t >, 2508
 - std::ctype_byname, 2525
 - std::ctype_byname< char >, 2539
 - do_transform
 - std::collate, 2447
 - std::collate_byname, 2453
 - do_truename
 - std::numpunct, 3116
-

- std::numpunct_byname, 3123
- do_widen
 - std::__ctype_abstract_base, 1398, 1399
 - std::ctype, 2476
 - std::ctype< char >, 2491
 - std::ctype< wchar_t >, 2509
 - std::ctype_byname, 2526
 - std::ctype_byname< char >, 2540
- duration_cast
 - std::chrono, 786
- dynamic_pointer_cast
 - std, 664
- eback
 - __gnu_cxx::enc_filebuf, 933
 - __gnu_cxx::stdio_filebuf, 1004
 - __gnu_cxx::stdio_sync_filebuf, 1038
 - std::basic_filebuf, 1622
 - std::basic_streambuf, 2174
 - std::basic_stringbuf, 2265
- ECMAScript
 - std::regex_constants, 804
- egptr
 - __gnu_cxx::enc_filebuf, 934
 - __gnu_cxx::stdio_filebuf, 1005
 - __gnu_cxx::stdio_sync_filebuf, 1039
 - std::basic_filebuf, 1622
 - std::basic_streambuf, 2175
 - std::basic_stringbuf, 2266
- egrep
 - std::regex_constants, 805
- element_type
 - std::auto_ptr, 1599
- ellint_1
 - tr1_math_spec_func, 122
- ellint_2
 - tr1_math_spec_func, 122
- ellint_3
 - tr1_math_spec_func, 122
- emplace
 - std::deque, 2575
 - std::list, 2844
 - std::vector, 3383
- emplace_after
 - std::forward_list, 2635
- emplace_front
 - std::forward_list, 2636
- empty
 - __gnu_cxx::__versa_string, 861
 - __gnu_debug::basic_string, 1129
 - std::basic_string, 2222
 - std::deque, 2575
 - std::forward_list, 2636
 - std::list, 2844
 - std::map, 2896
 - std::match_results, 2919
 - std::multimap, 2992
 - std::multiset, 3013
 - std::priority_queue, 3171
 - std::queue, 3178
 - std::set, 3239
 - std::stack, 3280
 - std::vector, 3383
- enc_filebuf.h, 3540
- end
 - __gnu_cxx::__versa_string, 861, 862
 - __gnu_cxx::temporary_buffer, 1065
 - __gnu_debug::basic_string, 1129
 - __gnu_parallel::_PseudoSequence, 1266
 - numeric_arrays, 96, 97
 - std, 665
 - std::_Temporary_buffer, 1564
 - std::basic_fstream, 1718
 - std::basic_ifstream, 1777
 - std::basic_ios, 1813
 - std::basic_iostream, 1884
 - std::basic_istream, 1941
 - std::basic_istreamstream, 2002
 - std::basic_ofstream, 2052
 - std::basic_ostream, 2100
 - std::basic_ostreamstream, 2150
 - std::basic_string, 2222
 - std::basic_stringstream, 2357
 - std::deque, 2576
 - std::forward_list, 2637
 - std::ios_base, 2754
 - std::list, 2844, 2845

- std::map, 2897
- std::match_results, 2920
- std::multimap, 2992, 2993
- std::multiset, 3013
- std::set, 3239
- std::vector, 3384
- endl
 - std, 666
- ends
 - std, 666
- eof
 - std::basic_fstream, 1668
 - std::basic_ifstream, 1738
 - std::basic_ios, 1795
 - std::basic_iostream, 1836
 - std::basic_istream, 1904
 - std::basic_istream, 1964
 - std::basic_ofstream, 2023
 - std::basic_ostream, 2072
 - std::basic_ostringstream, 2122
 - std::basic_stringstream, 2308
- eofbit
 - std::basic_fstream, 1718
 - std::basic_ifstream, 1777
 - std::basic_ios, 1813
 - std::basic_iostream, 1884
 - std::basic_istream, 1941
 - std::basic_istream, 2002
 - std::basic_ofstream, 2053
 - std::basic_ostream, 2100
 - std::basic_ostringstream, 2150
 - std::basic_stringstream, 2357
 - std::ios_base, 2755
- epptr
 - __gnu_cxx::enc_filebuf, 934
 - __gnu_cxx::stdio_filebuf, 1005
 - __gnu_cxx::stdio_sync_filebuf, 1039
 - std::basic_filebuf, 1623
 - std::basic_streambuf, 2175
 - std::basic_stringbuf, 2266
- epsilon
 - std::numeric_limits, 3080
- equal
 - non_mutating_algorithms, 158, 159
 - std::istreambuf_iterator, 2816
- equal_range
 - binary_search_algorithms, 203, 204
 - std::map, 2897, 2898
 - std::multimap, 2993
 - std::multiset, 3013, 3014
 - std::set, 3239, 3240
- equally_split
 - __gnu_parallel, 466
- equally_split.h, 3541
- equally_split_point
 - __gnu_parallel, 466
- erase
 - __gnu_cxx::__versa_string, 862, 863
 - __gnu_debug::basic_string, 1130, 1131
 - std::basic_string, 2222, 2223
 - std::deque, 2576, 2577
 - std::list, 2845
 - std::map, 2898, 2899
 - std::multimap, 2994, 2995
 - std::multiset, 3015, 3016
 - std::set, 3240, 3241
 - std::vector, 3385
- erase_after
 - std::forward_list, 2637, 2638
- error_backref
 - std::regex_constants, 802
- error_badbrace
 - std::regex_constants, 802
- error_badrepeat
 - std::regex_constants, 802
- error_brace
 - std::regex_constants, 802
- error_brack
 - std::regex_constants, 803
- error_collate
 - std::regex_constants, 803
- error_complexity
 - std::regex_constants, 803
- error_constants.h, 3541
- error_ctype
 - std::regex_constants, 803
- error_escape
 - std::regex_constants, 803
- error_paren

- std::regex_constants, 803
- error_range
 - std::regex_constants, 803
- error_space
 - std::regex_constants, 803
- error_stack
 - std::regex_constants, 804
- error_type
 - std::regex_constants, 802
- event
 - std::basic_fstream, 1664
 - std::basic_ifstream, 1735
 - std::basic_ios, 1792
 - std::basic_iostream, 1833
 - std::basic_istream, 1901
 - std::basic_istream, 1961
 - std::basic_ofstream, 2019
 - std::basic_ostream, 2069
 - std::basic_ostringstream, 2118
 - std::basic_stringstream, 2304
 - std::ios_base, 2745
- event_callback
 - std::basic_fstream, 1659
 - std::basic_ifstream, 1731
 - std::basic_ios, 1788
 - std::basic_iostream, 1828
 - std::basic_istream, 1897
 - std::basic_istream, 1957
 - std::basic_ofstream, 2015
 - std::basic_ostream, 2065
 - std::basic_ostringstream, 2114
 - std::basic_stringstream, 2299
 - std::ios_base, 2742
- exception, 3542
- exception.hpp, 3543
- exception_defines.h, 3544
- exception_ptr.h, 3544
- Exceptions, 31
- exceptions
 - __verbose_terminate_handler, 34
 - copy_exception, 34
 - current_exception, 34
 - make_exception_ptr, 34
 - rethrow_exception, 35
 - rethrow_if_nested, 35
 - set_terminate, 35
 - set_unexpected, 35
 - std::basic_fstream, 1669
 - std::basic_ifstream, 1739
 - std::basic_ios, 1796
 - std::basic_iostream, 1836, 1837
 - std::basic_istream, 1904, 1905
 - std::basic_istream, 1964, 1965
 - std::basic_ofstream, 2024
 - std::basic_ostream, 2072, 2073
 - std::basic_ostringstream, 2122, 2123
 - std::basic_stringstream, 2308, 2309
 - terminate, 35
 - terminate_handler, 33
 - throw_with_nested, 36
 - uncaught_exception, 36
 - unexpected, 36
 - unexpected_handler, 33
- exp
 - complex_numbers, 44
- expint
 - tr1_math_spec_func, 123
- exponential_distribution
 - std::exponential_distribution, 2611
- extc++.h, 3545
- extended
 - std::regex_constants, 805
- Extensions, 9
- extptr_allocator.h, 3545
- fabs
 - complex_numbers, 45
 - std, 666
- facet
 - std::locale::facet, 2872
- fail
 - std::basic_fstream, 1670
 - std::basic_ifstream, 1740
 - std::basic_ios, 1797
 - std::basic_iostream, 1838
 - std::basic_istream, 1905
 - std::basic_istream, 1965
 - std::basic_ofstream, 2025
 - std::basic_ostream, 2073
 - std::basic_ostringstream, 2123
 - std::basic_stringstream, 2309
- failbit

-
- std::basic_fstream, 1718
 - std::basic_ifstream, 1778
 - std::basic_ios, 1813
 - std::basic_iostream, 1885
 - std::basic_istream, 1942
 - std::basic_istream, 2003
 - std::basic_ofstream, 2053
 - std::basic_ostream, 2100
 - std::basic_ostringstream, 2151
 - std::basic_stringstream, 2357
 - std::ios_base, 2755
 - failed
 - std::ostreambuf_iterator, 3135
 - false_type
 - metaprogramming, 300
 - falsename
 - std::numpunct, 3117
 - std::numpunct_byname, 3124
 - fd
 - __gnu_cxx::stdio_filebuf, 1006
 - features.h, 3546
 - _GLIBCXX_BAL_QUICKSORT, 3546
 - _GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS, 3546
 - _GLIBCXX_FIND_EQUAL_SPLIT, 3547
 - _GLIBCXX_FIND_GROWING_BLOCKS, 3547
 - _GLIBCXX_MERGESORT, 3547
 - _GLIBCXX_QUICKSORT, 3547
 - _GLIBCXX_TREE_DYNAMIC_BALANCING, 3548
 - _GLIBCXX_TREE_FULL_COPY, 3548
 - _GLIBCXX_TREE_INITIAL_SPLITTING, 3548
 - fenv.h, 3549
 - file
 - __gnu_cxx::stdio_filebuf, 1006
 - __gnu_cxx::stdio_sync_filebuf, 1040
 - filebuf
 - io, 63
 - fill
 - mutating_algorithms, 134
 - std::basic_fstream, 1670, 1671
 - std::basic_ifstream, 1740, 1741
 - std::basic_ios, 1797
 - std::basic_iostream, 1838
 - std::basic_istream, 1906
 - std::basic_istream, 1966
 - std::basic_ofstream, 2025, 2026
 - std::basic_ostream, 2074
 - std::basic_ostringstream, 2124
 - std::basic_stringstream, 2310
 - fill_minimal_n
 - __gnu_parallel::_Settings, 1279
 - fill_n
 - mutating_algorithms, 134
 - find
 - __gnu_cxx::__versa_string, 864, 865
 - __gnu_debug::basic_string, 1131, 1132
 - non_mutating_algorithms, 159
 - std::basic_string, 2224, 2225
 - std::map, 2900
 - std::multimap, 2996
 - std::multiset, 3016, 3017
 - std::set, 3242
 - find.h, 3549
 - find_end
 - non_mutating_algorithms, 160, 161
 - find_first_not_of
 - __gnu_cxx::__versa_string, 866–868
 - __gnu_debug::basic_string, 1133, 1134
 - std::basic_string, 2226–2228
 - find_first_of
 - __gnu_cxx::__versa_string, 868–870
 - __gnu_debug::basic_string, 1135, 1136
 - non_mutating_algorithms, 161, 162
 - std::basic_string, 2228–2230
 - find_if
 - non_mutating_algorithms, 163
 - find_if_not
 - non_mutating_algorithms, 163
 - find_increasing_factor
-

-
- `__gnu_parallel::_Settings`, 1279
 - `find_initial_block_size`
 - `__gnu_parallel::_Settings`, 1279
 - `find_last_not_of`
 - `__gnu_cxx::__versa_string`, 871–873
 - `__gnu_debug::basic_string`, 1136–1138
 - `std::basic_string`, 2230–2232
 - `find_last_of`
 - `__gnu_cxx::__versa_string`, 873–875
 - `__gnu_debug::basic_string`, 1138–1140
 - `std::basic_string`, 2232–2234
 - `find_maximum_block_size`
 - `__gnu_parallel::_Settings`, 1279
 - `find_scale_factor`
 - `__gnu_parallel::_Settings`, 1279
 - `find_selectors.h`, 3550
 - `find_sequential_search_size`
 - `__gnu_parallel::_Settings`, 1280
 - `first`
 - `__gnu_parallel::_IteratorPair`, 1221
 - `std::pair`, 3145
 - `std::sub_match`, 3291
 - `first_argument_type`
 - `__gnu_cxx::project1st`, 975
 - `__gnu_cxx::project2nd`, 976
 - `__gnu_parallel::_EqualFromLess`, 1214
 - `__gnu_parallel::_EqualTo`, 1216
 - `__gnu_parallel::_Less`, 1226
 - `__gnu_parallel::_Lexicographic`, 1228
 - `__gnu_parallel::_-`
 - `LexicographicReverse`, 1230
 - `__gnu_parallel::_Multiplies`, 1258
 - `__gnu_parallel::_Plus`, 1262
 - `std::_Maybe_unary_or_binary_-function< _Res, _T1, _T2 >`, 1551
 - `std::binary_function`, 2368
 - `std::binary_negate`, 2369
 - `std::const_mem_fun1_ref_t`, 2461
 - `std::const_mem_fun1_t`, 2463
 - `std::divides`, 2600
 - `std::equal_to`, 2605
 - `std::greater`, 2686
 - `std::greater_equal`, 2688
 - `std::less`, 2823
 - `std::less_equal`, 2825
 - `std::logical_and`, 2878
 - `std::logical_or`, 2882
 - `std::mem_fun1_ref_t`, 2926
 - `std::mem_fun1_t`, 2928
 - `std::minus`, 2942
 - `std::modulus`, 2944
 - `std::multiplies`, 3006
 - `std::not_equal_to`, 3042
 - `std::owner_less< shared_ptr< _Tp > >`, 3140
 - `std::owner_less< weak_ptr< _Tp > >`, 3141
 - `std::plus`, 3159
 - `std::pointer_to_binary_function`, 3161
 - `fixed`
 - `std`, 667
 - `std::basic_fstream`, 1719
 - `std::basic_ifstream`, 1778
 - `std::basic_ios`, 1814
 - `std::basic_iostream`, 1885
 - `std::basic_istream`, 1942
 - `std::basic_istreamstream`, 2003
 - `std::basic_ofstream`, 2053
 - `std::basic_ostream`, 2101
 - `std::basic_ostreamstream`, 2151
 - `std::basic_stringstream`, 2358
 - `std::ios_base`, 2755
 - `flags`
 - `std::basic_fstream`, 1671, 1672
 - `std::basic_ifstream`, 1741, 1742
 - `std::basic_ios`, 1798
 - `std::basic_iostream`, 1839
 - `std::basic_istream`, 1906, 1907
 - `std::basic_istreamstream`, 1967
 - `std::basic_ofstream`, 2026
 - `std::basic_ostream`, 2075
 - `std::basic_ostreamstream`, 2125
 - `std::basic_regex`, 2165
 - `std::basic_stringstream`, 2311
-

-
- std::ios_base, 2746
 - flip
 - std::bitset, 2387
 - float_denorm_style
 - std, 640
 - float_round_style
 - std, 641
 - floatfield
 - std::basic_fstream, 1719
 - std::basic_ifstream, 1779
 - std::basic_ios, 1814
 - std::basic_iostream, 1886
 - std::basic_istream, 1943
 - std::basic_istream, 2004
 - std::basic_ofstream, 2054
 - std::basic_ostream, 2101
 - std::basic_ostringstream, 2151
 - std::basic_stringstream, 2358
 - std::ios_base, 2756
 - flush
 - std, 667
 - std::basic_fstream, 1672
 - std::basic_iostream, 1840
 - std::basic_ofstream, 2027
 - std::basic_ostream, 2075
 - std::basic_ostringstream, 2125
 - std::basic_stringstream, 2311
 - fmtflags
 - std::basic_fstream, 1660
 - std::basic_ifstream, 1731
 - std::basic_ios, 1788
 - std::basic_iostream, 1828
 - std::basic_istream, 1897
 - std::basic_istream, 1957
 - std::basic_ofstream, 2016
 - std::basic_ostream, 2065
 - std::basic_ostringstream, 2115
 - std::basic_stringstream, 2300
 - std::ios_base, 2742
 - for_each
 - non_mutating_algorithms, 164
 - for_each.h, 3550
 - for_each_minimal_n
 - __gnu_parallel::_Settings, 1280
 - for_each_selectors.h, 3551
 - format
 - std::match_results, 2920
 - format_default
 - std::regex_constants, 805
 - format_first_only
 - std::regex_constants, 806
 - format_no_copy
 - std::regex_constants, 806
 - format_sed
 - std::regex_constants, 806
 - formatter.h, 3553
 - forward
 - std, 667
 - forward_list, 3554, 3555
 - std::forward_list, 2629–2631
 - forward_list.h, 3556
 - forward_list.tcc, 3558
 - fpos
 - std::fpos, 2650
 - frac_digits
 - std::moneypunct, 2967
 - std::moneypunct_byname, 2979
 - front
 - __gnu_cxx::__versa_string, 876
 - __gnu_debug::basic_string, 1140
 - std::basic_string, 2234
 - std::deque, 2577
 - std::forward_list, 2638
 - std::list, 2846
 - std::queue, 3178
 - std::vector, 3386
 - front_insert_iterator
 - std::front_insert_iterator, 2655
 - front_inserter
 - iterators, 289
 - fstream, 3558
 - io, 63
 - fstream.tcc, 3559
 - functexcept.h, 3559
 - function
 - std::function<_Res(_-ArgTypes...)>, 2658–2660
 - Function Objects, 268
 - functional, 3560, 3565
 - functional_hash.h, 3567
 - functions.h, 3567
 - functors
-

- mem_fn, 270
- future, 3570
 - std::future, 2666
 - std::future< _Res & >, 2669
 - std::future< void >, 2672
- future_category
 - futures, 59
- future_errc
 - futures, 58
- future_status
 - futures, 58
- Futures, 56
- futures
 - async, 59
 - future_category, 59
 - future_errc, 58
 - future_status, 58
 - launch, 58
 - make_error_code, 59
 - make_error_condition, 59
 - swap, 60
- gamma_distribution
 - std::gamma_distribution, 2677
- gbump
 - __gnu_cxx::enc_filebuf, 934
 - __gnu_cxx::stdio_filebuf, 1006
 - __gnu_cxx::stdio_sync_filebuf, 1040
 - std::basic_filebuf, 1623
 - std::basic_streambuf, 2176
 - std::basic_stringbuf, 2267
- gcount
 - std::basic_fstream, 1672
 - std::basic_ifstream, 1742
 - std::basic_iostream, 1840
 - std::basic_istream, 1907
 - std::basic_istreamstream, 1967
 - std::basic_stringstream, 2312
- generate
 - mutating_algorithms, 135
- generate_canonical
 - random, 229
- generate_minimal_n
 - __gnu_parallel::_Settings, 1280
- generate_n
 - mutating_algorithms, 135
- get
 - __gnu_parallel::_Settings, 1277
 - std::auto_ptr, 1601
 - std::basic_fstream, 1673–1676
 - std::basic_ifstream, 1742–1745
 - std::basic_iostream, 1840–1843
 - std::basic_istream, 1907–1910
 - std::basic_istreamstream, 1968–1971
 - std::basic_stringstream, 2312–2315
 - std::future, 2667
 - std::future< _Res & >, 2670
 - std::future< void >, 2673
 - std::money_get, 2950
 - std::num_get, 3053–3062
 - std::shared_future, 3252
 - std::shared_future< _Res & >, 3255
- get_allocator
 - __gnu_cxx::__versa_string, 876
 - __gnu_debug::basic_string, 1140
 - std::basic_string, 2235
 - std::deque, 2578
 - std::forward_list, 2639
 - std::list, 2846
 - std::map, 2901
 - std::match_results, 2920
 - std::multimap, 2997
 - std::multiset, 3017
 - std::set, 3243
- get_date
 - std::time_get, 3304
 - std::time_get_byname, 3315
- get_deleter
 - pointer_abstractions, 72
- get_id
 - std::this_thread, 811
- get_money
 - std, 667
- get_monthname
 - std::time_get, 3304
 - std::time_get_byname, 3316
- get_temporary_buffer
 - std, 668
- get_time
 - std::time_get, 3305
 - std::time_get_byname, 3317

- get_weekday
 - std::time_get, 3306
 - std::time_get_byname, 3317
- get_year
 - std::time_get, 3307
 - std::time_get_byname, 3318
- getline
 - std, 668–670
 - std::basic_fstream, 1676, 1677
 - std::basic_ifstream, 1746
 - std::basic_iostream, 1844, 1845
 - std::basic_istream, 1911, 1912
 - std::basic_istreamstream, 1971, 1972
 - std::basic_stringstream, 2316, 2317
- getloc
 - __gnu_cxx::enc_filebuf, 935
 - __gnu_cxx::stdio_filebuf, 1007
 - __gnu_cxx::stdio_sync_filebuf, 1041
 - std::basic_filebuf, 1624
 - std::basic_fstream, 1678
 - std::basic_ifstream, 1747
 - std::basic_ios, 1799
 - std::basic_iostream, 1845
 - std::basic_istream, 1912
 - std::basic_istreamstream, 1973
 - std::basic_ofstream, 2027
 - std::basic_ostream, 2076
 - std::basic_ostreamstream, 2126
 - std::basic_regex, 2165
 - std::basic_streambuf, 2176
 - std::basic_stringbuf, 2267
 - std::basic_stringstream, 2317
 - std::ios_base, 2746
 - std::regex_traits, 3209
- global
 - std::locale, 2864
- good
 - std::basic_fstream, 1678
 - std::basic_ifstream, 1748
 - std::basic_ios, 1799
 - std::basic_iostream, 1846
 - std::basic_istream, 1913
 - std::basic_istreamstream, 1973
 - std::basic_ofstream, 2028
 - std::basic_ostream, 2076
 - std::basic_ostreamstream, 2126
 - std::basic_stringstream, 2318
- goodbit
 - std::basic_fstream, 1719
 - std::basic_ifstream, 1779
 - std::basic_ios, 1814
 - std::basic_iostream, 1886
 - std::basic_istream, 1943
 - std::basic_istreamstream, 2004
 - std::basic_ofstream, 2054
 - std::basic_ostream, 2101
 - std::basic_ostreamstream, 2152
 - std::basic_stringstream, 2358
 - std::ios_base, 2756
- gptr
 - __gnu_cxx::enc_filebuf, 935
 - __gnu_cxx::stdio_filebuf, 1007
 - __gnu_cxx::stdio_sync_filebuf, 1041
 - std::basic_filebuf, 1624
 - std::basic_streambuf, 2177
 - std::basic_stringbuf, 2268
- grep
 - std::regex_constants, 806
- grouping
 - std::moneypunct, 2968
 - std::moneypunct_byname, 2980
 - std::numpunct, 3117
 - std::numpunct_byname, 3124
- gslice
 - numeric_arrays, 91
- gslice.h, 3572
- gslice_array
 - numeric_arrays, 91
- gslice_array.h, 3573
- has_denorm
 - std::__numeric_limits_base, 1496
 - std::numeric_limits, 3082
- has_denorm_loss
 - std::__numeric_limits_base, 1496
 - std::numeric_limits, 3082
- has_facet
 - std, 671
 - std::locale, 2867
 - std::locale::id, 2873

-
- has_infinity
 - std::__numeric_limits_base, 1496
 - std::numeric_limits, 3083
 - has_quiet_NaN
 - std::__numeric_limits_base, 1496
 - std::numeric_limits, 3083
 - has_signaling_NaN
 - std::__numeric_limits_base, 1496
 - std::numeric_limits, 3083
 - hash
 - std::collate, 2447
 - std::collate_byname, 2453
 - hash_bytes.h, 3573
 - hash_fun.h, 3574
 - hash_map, 3574
 - hash_policy.hpp, 3575
 - hash_set, 3576
 - Hashes, 225
 - hashtable.h, 3577, 3578
 - hashtable_policy.h, 3578
 - Heap, 277
 - heap_algorithms
 - is_heap, 279
 - is_heap_until, 279, 280
 - make_heap, 280, 281
 - pop_heap, 281, 282
 - push_heap, 282
 - sort_heap, 283
 - hermite
 - tr1_math_spec_func, 123
 - hex
 - std, 671
 - std::basic_fstream, 1720
 - std::basic_ifstream, 1779
 - std::basic_ios, 1815
 - std::basic_iostream, 1886
 - std::basic_istream, 1943
 - std::basic_istreamstream, 2004
 - std::basic_ofstream, 2054
 - std::basic_ostream, 2102
 - std::basic_ostreamstream, 2152
 - std::basic_stringstream, 2359
 - std::ios_base, 2756
 - hours
 - std::chrono, 785
 - hyperg
 - tr1_math_spec_func, 123
 - I/O, 60
 - icase
 - std::regex_constants, 806
 - id
 - std::collate, 2448
 - std::collate_byname, 2454
 - std::ctype, 2483
 - std::ctype< char >, 2498
 - std::ctype< wchar_t >, 2516
 - std::ctype_byname, 2533
 - std::ctype_byname< char >, 2547
 - std::locale::id, 2873
 - std::messages, 2936
 - std::messages_byname, 2940
 - std::money_get, 2951
 - std::money_put, 2958
 - std::moneypunct, 2971
 - std::moneypunct_byname, 2983
 - std::num_get, 3063
 - std::num_put, 3078
 - std::numpunct, 3119
 - std::numpunct_byname, 3126
 - std::time_get, 3308
 - std::time_get_byname, 3319
 - std::time_put, 3324
 - std::time_put_byname, 3329
 - identity_element
 - SGIextensions, 19
 - ifstream
 - io, 63
 - ignore
 - std::basic_fstream, 1678–1680
 - std::basic_ifstream, 1748, 1749
 - std::basic_iostream, 1846, 1847
 - std::basic_istream, 1913, 1914
 - std::basic_istreamstream, 1973–1975
 - std::basic_stringstream, 2318, 2319
 - imbue
 - __gnu_cxx::enc_filebuf, 935
 - __gnu_cxx::stdio_filebuf, 1008
 - __gnu_cxx::stdio_sync_filebuf, 1042
 - std::basic_filebuf, 1625
 - std::basic_fstream, 1680
-

- std::basic_ifstream, [1750](#)
- std::basic_ios, [1799](#)
- std::basic_iostream, [1848](#)
- std::basic_istream, [1915](#)
- std::basic_istreamstream, [1975](#)
- std::basic_ofstream, [2028](#)
- std::basic_ostream, [2076](#)
- std::basic_ostringstream, [2126](#)
- std::basic_regex, [2165](#)
- std::basic_streambuf, [2177](#)
- std::basic_stringbuf, [2269](#)
- std::basic_stringstream, [2320](#)
- std::ios_base, [2747](#)
- std::regex_traits, [3209](#)
- in
 - std::__codecvt_abstract_base, [1387](#)
 - std::basic_fstream, [1720](#)
 - std::basic_ifstream, [1779](#)
 - std::basic_ios, [1815](#)
 - std::basic_iostream, [1886](#)
 - std::basic_istream, [1943](#)
 - std::basic_istreamstream, [2004](#)
 - std::basic_ofstream, [2055](#)
 - std::basic_ostream, [2102](#)
 - std::basic_ostringstream, [2152](#)
 - std::basic_stringstream, [2359](#)
 - std::codecvt, [2415](#)
 - std::codecvt< _InternT, _ExternT, encoding_state >, [2421](#)
 - std::codecvt< char, char, mbstate_t >, [2426](#)
 - std::codecvt< wchar_t, char, mbstate_t >, [2431](#)
 - std::codecvt_byname, [2438](#)
 - std::ios_base, [2757](#)
- in_avail
 - __gnu_cxx::enc_filebuf, [936](#)
 - __gnu_cxx::stdio_filebuf, [1009](#)
 - __gnu_cxx::stdio_sync_filebuf, [1042](#)
 - std::basic_filebuf, [1625](#)
 - std::basic_streambuf, [2178](#)
 - std::basic_stringbuf, [2269](#)
- include/ Directory Reference, [351](#)
- include/backward/ Directory Reference, [332](#)
- include/bits/ Directory Reference, [336](#)
- include/debug/ Directory Reference, [340](#)
- include/decimal/ Directory Reference, [341](#)
- include/ext/ Directory Reference, [345](#)
- include/ext/pb_ds/ Directory Reference, [359](#)
- include/ext/pb_ds/detail/ Directory Reference, [342](#)
- include/parallel/ Directory Reference, [356](#)
- include/profile/ Directory Reference, [361](#)
- include/profile/impl/ Directory Reference, [348](#)
- include/tr1/ Directory Reference, [363](#)
- include/x86_64-unknown-linux-gnu/ Directory Reference, [364](#)
- include/x86_64-unknown-linux-gnu/bits/ Directory Reference, [334](#)
- includes
 - set_algorithms, [194](#), [195](#)
- increment
 - std::linear_congruential_engine, [2832](#)
- independent_bits_engine
 - std::independent_bits_engine, [2722](#)–[2724](#)
- indirect_array
 - numeric_arrays, [92](#)
- indirect_array.h, [3579](#)
- infinity
 - std::numeric_limits, [3080](#)
- init
 - std::basic_fstream, [1681](#)
 - std::basic_ifstream, [1750](#)
 - std::basic_ios, [1800](#)
 - std::basic_iostream, [1848](#)
 - std::basic_istream, [1915](#)
 - std::basic_istreamstream, [1976](#)
 - std::basic_ofstream, [2029](#)
 - std::basic_ostream, [2077](#)
 - std::basic_ostringstream, [2127](#)
 - std::basic_stringstream, [2320](#)
- initializer_list, [3580](#)
- inner_product
 - std, [672](#)
- inplace_merge

- 3914

- iostate
 - std::basic_fstream, 1661
 - std::basic_ifstream, 1732
 - std::basic_ios, 1790
 - std::basic_iostream, 1830
 - std::basic_istream, 1899
 - std::basic_istream, 1958
 - std::basic_ofstream, 2017
 - std::basic_ostream, 2066
 - std::basic_ostringstream, 2116
 - std::basic_stringstream, 2301
 - std::ios_base, 2743
- iostream, 3586
 - io, 64
- iota
 - SGIextensions, 19
 - std, 673
- is
 - std::__ctype_abstract_base, 1400
 - std::ctype, 2477, 2478
 - std::ctype< char >, 2492
 - std::ctype< wchar_t >, 2510
 - std::ctype_byname, 2527, 2528
 - std::ctype_byname< char >, 2541
- is_bounded
 - std::__numeric_limits_base, 1497
 - std::numeric_limits, 3083
- is_exact
 - std::__numeric_limits_base, 1497
 - std::numeric_limits, 3083
- is_heap
 - heap_algorithms, 279
 - SGIextensions, 19, 20
- is_heap_until
 - heap_algorithms, 279, 280
- is_iec559
 - std::__numeric_limits_base, 1497
 - std::numeric_limits, 3083
- is_integer
 - std::__numeric_limits_base, 1497
 - std::numeric_limits, 3084
- is_modulo
 - std::__numeric_limits_base, 1497
 - std::numeric_limits, 3084
- is_open
 - __gnu_cxx::enc_filebuf, 936
 - __gnu_cxx::stdio_filebuf, 1009
 - std::basic_filebuf, 1626
 - std::basic_fstream, 1681
 - std::basic_ifstream, 1751
 - std::basic_ofstream, 2029
- is_partitioned
 - mutating_algorithms, 136
- is_permutation
 - non_mutating_algorithms, 164, 165
- is_signed
 - std::__numeric_limits_base, 1497
 - std::numeric_limits, 3084
- is_sorted
 - SGIextensions, 20
 - sorting_algorithms, 174
- is_sorted_until
 - sorting_algorithms, 175
- is_specialized
 - std::__numeric_limits_base, 1498
 - std::numeric_limits, 3084
- isalnum
 - std, 674
- isalpha
 - std, 674
- iscntrl
 - std, 674
- isctype
 - regex, 237
- isdigit
 - std, 674
- isgraph
 - std, 674
- islower
 - std, 674
- isprint
 - std, 675
- ispunct
 - std, 675
- isspace
 - std, 675
- istream, 3586
 - io, 64
- istream.tcc, 3588
- istream_iterator
 - std::istream_iterator, 2811
- istream_type

- std::istreambuf_iterator, 2813
- istreambuf_iterator
 - std::istreambuf_iterator, 2815
- istreamstring
 - io, 64
- isupper
 - std, 675
- isxdigit
 - std, 675
- iter_swap
 - mutating_algorithms, 136
- iter_type
 - std::money_get, 2948
 - std::money_put, 2954
 - std::num_get, 3045
 - std::num_put, 3066
 - std::time_get, 3298
 - std::time_get_byname, 3310
 - std::time_put, 3321
 - std::time_put_byname, 3327
- iterator, 3588, 3589
 - std::set, 3232
- Iterator Tags, 291
- iterator.h, 3589
- iterator_category
 - std::back_insert_iterator, 1605
 - std::front_insert_iterator, 2654
 - std::insert_iterator, 2734
 - std::istream_iterator, 2810
 - std::istreambuf_iterator, 2814
 - std::iterator, 2818
 - std::ostream_iterator, 3129
 - std::ostreambuf_iterator, 3133
 - std::raw_storage_iterator, 3191
 - std::reverse_iterator, 3219
- iterator_tracker.h, 3590
- Iterators, 284
- iterators
 - __iterator_category, 288
 - back_inserter, 289
 - front_inserter, 289
 - inserter, 289
 - operator==, 290, 291
- iword
 - std::basic_fstream, 1681
 - std::basic_ifstream, 1751
 - std::basic_ios, 1800
 - std::basic_iostream, 1849
 - std::basic_istream, 1916
 - std::basic_istreamstring, 1976
 - std::basic_ofstream, 2029
 - std::basic_ostream, 2077
 - std::basic_ostringstream, 2127
 - std::basic_stringstream, 2321
 - std::ios_base, 2747
- k
 - std::negative_binomial_distribution, 3029
- key_comp
 - std::map, 2903
 - std::multimap, 2999
 - std::multiset, 3019
 - std::set, 3245
- key_compare
 - std::set, 3232
- key_type
 - std::set, 3232
- kill_dependency
 - atomics, 224
- L1_cache_size
 - __gnu_parallel::_Settings, 1280
- L2_cache_size
 - __gnu_parallel::_Settings, 1280
- laguerre
 - tr1_math_spec_func, 123
- lambda
 - std::exponential_distribution, 2611
- launch
 - futures, 58
- left
 - std, 675
 - std::basic_fstream, 1721
 - std::basic_ifstream, 1780
 - std::basic_ios, 1816
 - std::basic_iostream, 1887
 - std::basic_istream, 1944
 - std::basic_istreamstring, 2005
 - std::basic_ofstream, 2055
 - std::basic_ostream, 2103
 - std::basic_ostringstream, 2153

- std::basic_stringstream, 2360
- std::ios_base, 2757
- legendre
 - tr1_math_spec_func, 123
- length
 - __gnu_cxx::__versa_string, 882
 - __gnu_debug::basic_string, 1146
 - std::basic_string, 2240
 - std::match_results, 2921
 - std::regex_traits, 3209
 - std::sub_match, 3290
- lexicographical_compare
 - sorting_algorithms, 176
- lexicographical_compare_3way
 - SGIextensions, 20
- limits, 3592
- linear_congruential_engine
 - std::linear_congruential_engine, 2827, 2828
- list, 3594, 3595
 - std::list, 2837–2839
- list.tcc, 3596
- list_partition
 - __gnu_parallel, 467
- list_partition.h, 3596
- list_update_policy.hpp, 3597
- locale, 3598
 - std::locale, 2862, 2863
- locale_classes.h, 3598
- locale_classes.tcc, 3599
- locale_facets.h, 3599
- locale_facets.tcc, 3602
- locale_facets_nonio.h, 3602
- locale_facets_nonio.tcc, 3604
- localefwd.h, 3604
- Locales, 225
- lock
 - mutexes, 74
- log
 - complex_numbers, 45
- log10
 - complex_numbers, 45
- logic_error
 - std::logic_error, 2876
- lookup_classname
 - std::regex_traits, 3210
- lookup_collatename
 - std::regex_traits, 3211
- losertree.h, 3605
- lower_bound
 - binary_search_algorithms, 204, 205
 - std::map, 2904
 - std::multimap, 2999, 3000
 - std::multiset, 3019, 3020
 - std::set, 3245, 3246
- lowest
 - std::numeric_limits, 3081
- macros.h, 3607
 - _GLIBCXX_DEBUG_VERIFY, 3610
 - __glibcxx_check_erase, 3608
 - __glibcxx_check_erase_after, 3608
 - __glibcxx_check_erase_range, 3608
 - __glibcxx_check_erase_range_after, 3608
 - __glibcxx_check_heap_pred, 3608
 - __glibcxx_check_insert, 3608
 - __glibcxx_check_insert_after, 3609
 - __glibcxx_check_insert_range, 3609
 - __glibcxx_check_insert_range_-after, 3609
 - __glibcxx_check_partitioned_lower, 3610
 - __glibcxx_check_partitioned_-lower_pred, 3610
 - __glibcxx_check_partitioned_-upper_pred, 3610
 - __glibcxx_check_sorted_pred, 3610
- make_error_code
 - futures, 59
- make_error_condition
 - futures, 59
- make_exception_ptr
 - exceptions, 34
- make_heap
 - heap_algorithms, 280, 281
- make_pair
 - std, 676
- make_shared
 - pointer_abstractions, 72
- malloc_allocator.h, 3611

-
- map, 3612
 - std::map, 2891–2893
 - map.h, 3612, 3613
 - mark_count
 - std::basic_regex, 2165
 - mask_array
 - numeric_arrays, 92
 - mask_array.h, 3614
 - match_any
 - std::regex_constants, 807
 - match_continuous
 - std::regex_constants, 807
 - match_default
 - std::regex_constants, 807
 - match_flag_type
 - std::regex_constants, 801
 - match_not_bol
 - std::regex_constants, 807
 - match_not_bow
 - std::regex_constants, 807
 - match_not_eol
 - std::regex_constants, 807
 - match_not_eow
 - std::regex_constants, 808
 - match_not_null
 - std::regex_constants, 808
 - match_prev_avail
 - std::regex_constants, 808
 - match_results
 - std::match_results, 2918
 - Mathematical Special Functions, 116
 - max
 - __gnu_parallel, 467
 - numeric_arrays, 97
 - sorting_algorithms, 177, 178
 - std::bernoulli_distribution, 2364
 - std::binomial_distribution, 2375
 - std::cauchy_distribution, 2396
 - std::chi_squared_distribution, 2405
 - std::discard_block_engine, 2590
 - std::discrete_distribution, 2595
 - std::exponential_distribution, 2611
 - std::extreme_value_distribution, 2616
 - std::fisher_f_distribution, 2620
 - std::gamma_distribution, 2677
 - std::geometric_distribution, 2682
 - std::independent_bits_engine, 2725
 - std::linear_congruential_engine, 2828
 - std::lognormal_distribution, 2884
 - std::negative_binomial_distribution, 3029
 - std::normal_distribution, 3036
 - std::numeric_limits, 3081
 - std::piecewise_constant_distribution, 3148
 - std::piecewise_linear_distribution, 3154
 - std::poisson_distribution, 3165
 - std::shuffle_order_engine, 3271
 - std::student_t_distribution, 3283
 - std::uniform_int_distribution, 3344
 - std::uniform_real_distribution, 3348
 - std::weibull_distribution, 3401
 - max_digits10
 - std::__numeric_limits_base, 1498
 - std::numeric_limits, 3084
 - max_element
 - sorting_algorithms, 178
 - max_element_minimal_n
 - __gnu_parallel::Settings, 1280
 - max_exponent
 - std::__numeric_limits_base, 1498
 - std::numeric_limits, 3085
 - max_exponent10
 - std::__numeric_limits_base, 1498
 - std::numeric_limits, 3085
 - max_size
 - __gnu_cxx::__versa_string, 882
 - __gnu_debug::basic_string, 1146
 - std::basic_string, 2240
 - std::deque, 2580
 - std::forward_list, 2641
 - std::list, 2849
 - std::map, 2905
 - std::match_results, 2921
 - std::multimap, 3000
 - std::multiset, 3020
 - std::set, 3246
 - std::vector, 3388
 - mean
-

- std::normal_distribution, 3036
- std::poisson_distribution, 3165
- mem_fn
 - functors, 270
- Memory, 68
- memory, 3615
- memory_order
 - atomics, 224
- merge
 - sorting_algorithms, 179, 180
 - std::forward_list, 2641, 2642
 - std::list, 2849, 2850
- merge.h, 3616
- merge_minimal_n
 - __gnu_parallel::Settings, 1281
- merge_oversampling
 - __gnu_parallel::Settings, 1281
- messages
 - std::locale, 2868
 - std::messages, 2935
- messages_members.h, 3617
- Metaprogramming, 294
- metaprogramming
 - _GLIBCXX_HAS_NESTED_-TYPE, 300
 - false_type, 300
 - true_type, 300
- microseconds
 - std::chrono, 785
- milliseconds
 - std::chrono, 785
- min
 - __gnu_parallel, 468
 - numeric_arrays, 97
 - sorting_algorithms, 180, 181
 - std::bernoulli_distribution, 2364
 - std::binomial_distribution, 2375
 - std::cauchy_distribution, 2396
 - std::chi_squared_distribution, 2405
 - std::discard_block_engine, 2591
 - std::discrete_distribution, 2595
 - std::exponential_distribution, 2611
 - std::extreme_value_distribution, 2617
 - std::fisher_f_distribution, 2620
 - std::gamma_distribution, 2677
 - std::geometric_distribution, 2683
 - std::independent_bits_engine, 2725
 - std::linear_congruential_engine, 2828
 - std::lognormal_distribution, 2884
 - std::negative_binomial_distribution, 3029
 - std::normal_distribution, 3036
 - std::numeric_limits, 3081
 - std::piecewise_constant_distribution, 3148
 - std::piecewise_linear_distribution, 3154
 - std::poisson_distribution, 3165
 - std::shuffle_order_engine, 3271
 - std::student_t_distribution, 3283
 - std::uniform_int_distribution, 3344
 - std::uniform_real_distribution, 3348
 - std::weibull_distribution, 3402
- min_element
 - sorting_algorithms, 181, 182
- min_element_minimal_n
 - __gnu_parallel::Settings, 1281
- min_exponent
 - std::__numeric_limits_base, 1498
 - std::numeric_limits, 3085
- min_exponent10
 - std::__numeric_limits_base, 1498
 - std::numeric_limits, 3085
- minmax
 - sorting_algorithms, 182, 183
- minmax_element
 - sorting_algorithms, 183, 184
- minstd_rand
 - random_generators, 303
- minstd_rand0
 - random_generators, 303
- minutes
 - std::chrono, 786
- mismatch
 - non_mutating_algorithms, 165, 166
- modulus
 - std::linear_congruential_engine, 2832
- monetary
 - std::locale, 2869

-
- money_get
 - std::money_get, 2948
 - money_put
 - std::money_put, 2954
 - money_punct
 - std::money_punct, 2961, 2962
 - move
 - mutating_algorithms, 137
 - move.h, 3617
 - move_backward
 - mutating_algorithms, 138
 - mt19937
 - random_generators, 303
 - mt19937_64
 - random_generators, 304
 - mt_allocator.h, 3618
 - multimap
 - std::multimap, 2988–2990
 - multimap.h, 3619, 3621
 - multiplier
 - std::linear_congruential_engine, 2832
 - multiseq_partition
 - __gnu_parallel, 468
 - multiseq_selection
 - __gnu_parallel, 469
 - multiseq_selection.h, 3622
 - multiset
 - std::multiset, 3009–3011
 - multiset.h, 3623, 3624
 - multiway_merge
 - __gnu_parallel, 469
 - multiway_merge.h, 3625
 - __GLIBCXX_PARALLEL_-LENGTH, 3630
 - multiway_merge_3_variant
 - __gnu_parallel, 471
 - multiway_merge_4_variant
 - __gnu_parallel, 472
 - multiway_merge_exact_splitting
 - __gnu_parallel, 473
 - multiway_merge_loser_tree
 - __gnu_parallel, 473
 - multiway_merge_loser_tree_sentinel
 - __gnu_parallel, 474
 - multiway_merge_loser_tree_unguarded
 - __gnu_parallel, 475
 - multiway_merge_minimal_k
 - __gnu_parallel::_Settings, 1281
 - multiway_merge_minimal_n
 - __gnu_parallel::_Settings, 1281
 - multiway_merge_oversampling
 - __gnu_parallel::_Settings, 1282
 - multiway_merge_sampling_splitting
 - __gnu_parallel, 475
 - multiway_merge_sentinels
 - __gnu_parallel, 476
 - multiway_mergesort.h, 3630
 - Mutating, 129
 - mutating_algorithms
 - copy, 132
 - copy_backward, 132
 - copy_if, 133
 - copy_n, 133
 - fill, 134
 - fill_n, 134
 - generate, 135
 - generate_n, 135
 - is_partitioned, 136
 - iter_swap, 136
 - move, 137
 - move_backward, 138
 - partition, 138
 - partition_copy, 139
 - partition_point, 139
 - random_shuffle, 140
 - remove, 141
 - remove_copy, 142
 - remove_copy_if, 142
 - remove_if, 143
 - replace, 143
 - replace_copy_if, 144
 - replace_if, 144
 - reverse, 145
 - reverse_copy, 145
 - rotate, 146
 - rotate_copy, 147
 - shuffle, 147
 - stable_partition, 148
 - swap, 149
 - swap_ranges, 149
 - transform, 150
-

- unique, 151
- unique_copy, 152
- mutex, 3632
- Mutexes, 73
- mutexes
 - call_once, 74
 - lock, 74
 - try_lock, 75
- name
 - std::locale, 2865
 - std::type_info, 3337
- nanoseconds
 - std::chrono, 786
- narrow
 - std::__ctype_abstract_base, 1401
 - std::basic_fstream, 1682
 - std::basic_ifstream, 1751
 - std::basic_ios, 1801
 - std::basic_istream, 1849
 - std::basic_istream, 1916
 - std::basic_istream, 1977
 - std::basic_ofstream, 2030
 - std::basic_ostream, 2078
 - std::basic_ostream, 2128
 - std::basic_stringstream, 2321
 - std::ctype, 2478, 2479
 - std::ctype< char >, 2493
 - std::ctype< wchar_t >, 2511
 - std::ctype_byname, 2528, 2529
 - std::ctype_byname< char >, 2542
- native_handle
 - std::thread, 3294
- neg_format
 - std::moneypunct, 2968
 - std::moneypunct_byname, 2980
- negative_sign
 - std::moneypunct, 2969
 - std::moneypunct_byname, 2981
- Negators, 273
- negators
 - not1, 274
 - not2, 274
- nested_exception.h, 3633
- new, 3634
 - operator delete, 3635, 3636
 - operator new, 3637, 3638
- new_allocator.h, 3639
- new_handler
 - std, 639
- next_permutation
 - sorting_algorithms, 184, 185
- noboolalpha
 - std, 676
- Non-Mutating, 153
- non_mutating_algorithms
 - adjacent_find, 156
 - all_of, 156
 - any_of, 157
 - count, 157
 - count_if, 158
 - equal, 158, 159
 - find, 159
 - find_end, 160, 161
 - find_first_of, 161, 162
 - find_if, 163
 - find_if_not, 163
 - for_each, 164
 - is_permutation, 164, 165
 - mismatch, 165, 166
 - none_of, 166
 - search, 167
 - search_n, 168, 169
- none
 - std::bitset, 2387
 - std::locale, 2869
- none_of
 - non_mutating_algorithms, 166
- norm
 - complex_numbers, 45
- Normal Distributions, 312
- normal_distribution
 - std::normal_distribution, 3036
- noshowbase
 - std, 676
- noshowpoint
 - std, 677
- noshowpos
 - std, 677
- noskipws
 - std, 677
- nosubs

-
- std::regex_constants, 808
 - not1
 - negators, 274
 - not2
 - negators, 274
 - nounitbuf
 - std, 677
 - nouppercase
 - std, 677
 - npos
 - __gnu_cxx::__versa_string, 903
 - __gnu_debug::basic_string, 1161
 - std::basic_string, 2259
 - nth_element
 - sorting_algorithms, 185, 186
 - nth_element_minimal_n
 - __gnu_parallel::_Settings, 1282
 - num_get
 - std::num_get, 3046
 - num_put
 - std::num_put, 3066
 - numeric, 3640, 3641
 - std::locale, 2869
 - Numeric Arrays, 81
 - numeric_arrays
 - ~gslice, 94
 - apply, 95
 - begin, 95, 96
 - cshift, 96
 - end, 96, 97
 - gslice, 91
 - gslice_array, 91
 - indirect_array, 92
 - mask_array, 92
 - max, 97
 - min, 97
 - operator<=, 102
 - operator>=, 107, 108
 - operator*=, 99
 - operator~, 113
 - operator^=, 112, 113
 - operator+, 100
 - operator+=, 100
 - operator-, 100
 - operator-=, 101
 - operator/=: 101, 102
 - operator=, 102–107
 - operator%=, 98
 - operator&=: 98, 99
 - resize, 114
 - shift, 114
 - size, 115
 - slice, 92
 - slice_array, 93
 - start, 115
 - stride, 115, 116
 - sum, 116
 - valarray, 93, 94
 - numeric_traits.h, 3644
 - numeric_fwd.h, 3645
 - Numerics, 76
 - num_punct
 - std::num_punct, 3113, 3114
 - oct
 - std, 678
 - std::basic_fstream, 1721
 - std::basic_ifstream, 1780
 - std::basic_ios, 1816
 - std::basic_iostream, 1887
 - std::basic_istream, 1944
 - std::basic_istreamstream, 2005
 - std::basic_ofstream, 2055
 - std::basic_ostream, 2103
 - std::basic_ostreamstream, 2153
 - std::basic_stringstream, 2360
 - std::ios_base, 2757
 - off_type
 - __gnu_cxx::enc_filebuf, 931
 - __gnu_cxx::stdio_filebuf, 1001
 - __gnu_cxx::stdio_sync_filebuf, 1037
 - std::basic_filebuf, 1619
 - std::basic_fstream, 1661, 1662
 - std::basic_ifstream, 1733
 - std::basic_ios, 1790
 - std::basic_iostream, 1830
 - std::basic_istream, 1899
 - std::basic_istreamstream, 1959
 - std::basic_ofstream, 2017
 - std::basic_ostream, 2067
 - std::basic_ostreamstream, 2116
-

-
- std::basic_streambuf, [2172](#)
 - std::basic_stringbuf, [2264](#)
 - std::basic_stringstream, [2301](#), [2302](#)
 - ofstream
 - io, [64](#)
 - omp_loop.h, [3647](#)
 - omp_loop_static.h, [3648](#)
 - open
 - __gnu_cxx::enc_filebuf, [936](#), [937](#)
 - __gnu_cxx::stdio_filebuf, [1009](#), [1010](#)
 - std::basic_filebuf, [1626](#), [1627](#)
 - std::basic_fstream, [1682](#), [1683](#)
 - std::basic_ifstream, [1752](#)
 - std::basic_ofstream, [2030](#), [2031](#)
 - openmode
 - std::basic_fstream, [1662](#)
 - std::basic_ifstream, [1733](#)
 - std::basic_ios, [1791](#)
 - std::basic_iostream, [1831](#)
 - std::basic_istream, [1899](#)
 - std::basic_istreamstream, [1959](#)
 - std::basic_ofstream, [2017](#)
 - std::basic_ostream, [2067](#)
 - std::basic_ostreamstream, [2116](#)
 - std::basic_stringstream, [2302](#)
 - std::ios_base, [2744](#)
 - operator_iterator
 - __gnu_debug::_Safe_iterator, [1087](#)
 - operator_RAlter
 - __gnu_parallel::_GuardedIterator, [1218](#)
 - operator_bool
 - std::basic_istream::sentry, [1949](#)
 - std::basic_ostream::sentry, [2107](#)
 - std::function< _Res(_ArgTypes...)>, [2660](#)
 - operator_delete
 - new, [3635](#), [3636](#)
 - operator_new
 - new, [3637](#), [3638](#)
 - operator_streamoff
 - std::fpos, [2650](#)
 - operator_string_type
 - std::sub_match, [3290](#)
 - operator_void_*
 - std::basic_fstream, [1683](#)
 - std::basic_ifstream, [1753](#)
 - std::basic_ios, [1801](#)
 - std::basic_iostream, [1850](#)
 - std::basic_istream, [1917](#)
 - std::basic_istreamstream, [1977](#)
 - std::basic_ofstream, [2031](#)
 - std::basic_ostream, [2078](#)
 - std::basic_ostreamstream, [2128](#)
 - std::basic_stringstream, [2322](#)
 - operator_<
 - __gnu_cxx, [392](#), [393](#)
 - __gnu_parallel::_GuardedIterator, [1219](#)
 - regex, [241–243](#)
 - std, [685–691](#)
 - std::multiset, [3024](#)
 - operator_<<
 - complex_numbers, [50](#)
 - pointer_abstractions, [72](#)
 - random_distributions_bernoulli, [319](#)
 - random_distributions_normal, [315](#)
 - random_distributions_poisson, [325](#), [326](#)
 - random_distributions_uniform, [310](#)
 - random_generators, [307](#)
 - regex, [244](#)
 - std, [691–699](#)
 - std::basic_fstream, [1684–1690](#)
 - std::basic_iostream, [1850–1857](#)
 - std::basic_ofstream, [2032–2039](#)
 - std::basic_ostream, [2079–2086](#)
 - std::basic_ostreamstream, [2129–2136](#)
 - std::basic_stringstream, [2322–2329](#)
 - std::binomial_distribution, [2378](#)
 - std::bitset, [2388](#)
 - std::chi_squared_distribution, [2406](#)
 - std::discard_block_engine, [2592](#)
 - std::discrete_distribution, [2597](#)
 - std::fisher_f_distribution, [2622](#)
 - std::gamma_distribution, [2679](#)
 - std::linear_congruential_engine, [2830](#)
 - std::lognormal_distribution, [2885](#)
 - std::negative_binomial_distribution, [3031](#)
-

3924

- std::regex_token_iterator, 3205
- std::reverse_iterator, 3221
- operator*=
 - complex_numbers, 47
 - numeric_arrays, 99
 - std::indirect_array, 2729
 - std::mask_array, 2912
 - std::slice_array, 3276
- operator~
 - numeric_arrays, 113
 - std::bitset, 2391
- operator^
 - std, 725
- operator^=
 - numeric_arrays, 112, 113
 - std::bitset, 2390
 - std::indirect_array, 2730
 - std::mask_array, 2913
 - std::slice_array, 3277
- operator()
 - __gnu_cxx::subtractive_rng, 1062
 - __gnu_parallel::__Nothing, 1259
 - __gnu_parallel::__RandomNumber, 1271, 1272
 - __gnu_parallel::__accumulate_-selector, 1163
 - __gnu_parallel::__adjacent_find_-selector, 1166
 - __gnu_parallel::__count_if_selector, 1171
 - __gnu_parallel::__count_selector, 1173
 - __gnu_parallel::__fill_selector, 1175
 - __gnu_parallel::__find_first_of_-selector, 1177
 - __gnu_parallel::__find_if_selector, 1179
 - __gnu_parallel::__for_each_-selector, 1181
 - __gnu_parallel::__generate_-selector, 1182
 - __gnu_parallel::__identity_selector, 1187
 - __gnu_parallel::__inner_product_-selector, 1189
 - __gnu_parallel::__mismatch_-selector, 1193
 - __gnu_parallel::__replace_if_-selector, 1199
 - __gnu_parallel::__replace_selector, 1201
 - __gnu_parallel::__transform1_-selector, 1203
 - __gnu_parallel::__transform2_-selector, 1205
 - std::bernoulli_distribution, 2364
 - std::binomial_distribution, 2376
 - std::cauchy_distribution, 2396
 - std::chi_squared_distribution, 2405
 - std::discard_block_engine, 2591
 - std::discrete_distribution, 2595
 - std::exponential_distribution, 2611
 - std::extreme_value_distribution, 2617
 - std::fisher_f_distribution, 2621
 - std::function<_Res(_ArgTypes...)>, 2660
 - std::gamma_distribution, 2678
 - std::geometric_distribution, 2683
 - std::independent_bits_engine, 2725
 - std::linear_congruential_engine, 2829
 - std::locale, 2865
 - std::lognormal_distribution, 2884
 - std::negative_binomial_distribution, 3030
 - std::normal_distribution, 3036, 3037
 - std::piecewise_constant_distribution, 3148
 - std::piecewise_linear_distribution, 3154
 - std::poisson_distribution, 3165
 - std::shuffle_order_engine, 3271
 - std::student_t_distribution, 3283
 - std::uniform_int_distribution, 3345
 - std::uniform_real_distribution, 3349
 - std::weibull_distribution, 3402
- operator+
 - __gnu_cxx, 389–391
 - complex_numbers, 47, 48
 - numeric_arrays, 100

-
- std, 683, 684
 - std::fpos, 2650
 - std::reverse_iterator, 3221
 - operator++
 - __gnu_debug::_Safe_iterator, 1088
 - __gnu_parallel::_GuardedIterator, 1218
 - std::back_insert_iterator, 1607
 - std::front_insert_iterator, 2655
 - std::insert_iterator, 2735, 2736
 - std::istreambuf_iterator, 2816
 - std::ostreambuf_iterator, 3136
 - std::regex_iterator, 3199, 3200
 - std::regex_token_iterator, 3205, 3206
 - std::reverse_iterator, 3222
 - operator+=
 - __gnu_cxx::__versa_string, 883, 884
 - __gnu_debug::basic_string, 1146, 1147
 - complex_numbers, 48
 - numeric_arrays, 100
 - std::basic_string, 2241, 2242
 - std::complex, 2456
 - std::fpos, 2651
 - std::indirect_array, 2729
 - std::mask_array, 2912
 - std::reverse_iterator, 3222
 - std::slice_array, 3276
 - operator-
 - complex_numbers, 48, 49
 - numeric_arrays, 100
 - std::fpos, 2651
 - std::reverse_iterator, 3223
 - operator->
 - __gnu_debug::_Safe_iterator, 1089
 - std::auto_ptr, 1601
 - std::regex_iterator, 3200
 - std::regex_token_iterator, 3206
 - std::reverse_iterator, 3224
 - operator--
 - __gnu_debug::_Safe_iterator, 1089
 - std::reverse_iterator, 3223
 - operator-=
 - complex_numbers, 49
 - operator/
 - complex_numbers, 49, 50
 - operator/=
 - complex_numbers, 50
 - numeric_arrays, 101, 102
 - std::indirect_array, 2730
 - std::mask_array, 2912
 - std::slice_array, 3277
 - operator=
 - __gnu_cxx::__versa_string, 884–886
 - __gnu_debug::_Safe_iterator, 1090
 - complex_numbers, 51
 - numeric_arrays, 102–107
 - std::auto_ptr, 1601, 1602
 - std::back_insert_iterator, 1607
 - std::basic_regex, 2166, 2167
 - std::basic_string, 2242, 2243
 - std::deque, 2580, 2581
 - std::forward_list, 2642, 2643
 - std::front_insert_iterator, 2655
 - std::function<_Res(_ArgTypes...)>, 2661–2663
 - std::insert_iterator, 2736
 - std::list, 2850, 2851
 - std::locale, 2866
 - std::map, 2905, 2906
 - std::match_results, 2921
 - std::multimap, 3000, 3001
 - std::multiset, 3021
 - std::ostream_iterator, 3131
 - std::ostreambuf_iterator, 3136
 - std::regex_iterator, 3200
 - std::regex_token_iterator, 3206
 - std::set, 3246, 3247
 - std::vector, 3389
 - operator==
 - __gnu_cxx, 394–396
 - complex_numbers, 51
 - numeric_arrays, 101
 - std::complex, 2456
 - std::fpos, 2651
 - std::indirect_array, 2730
 - std::mask_array, 2912
 - std::reverse_iterator, 3224
 - std::slice_array, 3276
-

- iterators, [290](#), [291](#)
- random_distributions_bernoulli, [320](#)
- random_distributions_normal, [316](#)
- random_distributions_poisson, [327](#), [328](#)
- random_distributions_uniform, [311](#)
- regex, [247–250](#)
- std, [703–710](#)
- std::binomial_distribution, [2378](#)
- std::bitset, [2389](#)
- std::chi_squared_distribution, [2406](#)
- std::discard_block_engine, [2592](#)
- std::fisher_f_distribution, [2622](#)
- std::gamma_distribution, [2679](#)
- std::independent_bits_engine, [2726](#)
- std::linear_congruential_engine, [2830](#)
- std::locale, [2866](#)
- std::lognormal_distribution, [2886](#)
- std::multiset, [3025](#)
- std::negative_binomial_distribution, [3031](#)
- std::normal_distribution, [3039](#)
- std::poisson_distribution, [3167](#)
- std::regex_iterator, [3200](#)
- std::regex_token_iterator, [3207](#)
- std::shuffle_order_engine, [3272](#)
- std::student_t_distribution, [3284](#)
- operator%=
 - numeric_arrays, [98](#)
 - std::indirect_array, [2729](#)
 - std::mask_array, [2911](#)
 - std::slice_array, [3276](#)
- operator&
 - std, [682](#)
- operator&=
 - numeric_arrays, [98](#), [99](#)
 - std::bitset, [2388](#)
 - std::indirect_array, [2729](#)
 - std::mask_array, [2911](#)
 - std::slice_array, [3276](#)
- optimize
 - std::regex_constants, [808](#)
- os_defines.h, [3649](#)
- ostream, [3649](#)
 - io, [64](#)
 - ostream.tcc, [3651](#)
 - ostream_insert.h, [3651](#)
 - ostream_iterator
 - std::ostream_iterator, [3130](#)
 - ostream_type
 - std::ostream_iterator, [3129](#)
 - std::ostreambuf_iterator, [3133](#)
 - ostreambuf_iterator
 - std::ostreambuf_iterator, [3135](#)
 - ostreamstring
 - io, [65](#)
- out
 - std::__codecvt_abstract_base, [1388](#)
 - std::basic_fstream, [1721](#)
 - std::basic_ifstream, [1780](#)
 - std::basic_ios, [1816](#)
 - std::basic_iostream, [1887](#)
 - std::basic_istream, [1944](#)
 - std::basic_istreamstream, [2005](#)
 - std::basic_ofstream, [2056](#)
 - std::basic_ostream, [2103](#)
 - std::basic_ostreamstream, [2153](#)
 - std::basic_stringstream, [2360](#)
 - std::codecvt, [2416](#)
 - std::codecvt< _InternT, _ExternT, encoding_state >, [2422](#)
 - std::codecvt< char, char, mbstate_t >, [2427](#)
 - std::codecvt< wchar_t, char, mbstate_t >, [2432](#)
 - std::codecvt_byname, [2439](#)
 - std::ios_base, [2758](#)
- overflow
 - __gnu_cxx::enc_filebuf, [937](#)
 - __gnu_cxx::stdio_filebuf, [1011](#)
 - __gnu_cxx::stdio_sync_filebuf, [1043](#)
 - std::basic_filebuf, [1627](#)
 - std::basic_streambuf, [2178](#)
 - std::basic_stringbuf, [2269](#)
- P
 - std::bernoulli_distribution, [2364](#)
 - std::binomial_distribution, [2376](#)
 - std::geometric_distribution, [2683](#)

-
- std::negative_binomial_distribution, 3030
 - pair
 - std::pair, 3144, 3145
 - par_loop.h, 3652
 - parallel.h, 3652
 - parallel_balanced
 - __gnu_parallel, 431
 - parallel_omp_loop
 - __gnu_parallel, 431
 - parallel_omp_loop_static
 - __gnu_parallel, 431
 - parallel_taskqueue
 - __gnu_parallel, 431
 - parallel_unbalanced
 - __gnu_parallel, 431
 - parallel_multiway_merge
 - __gnu_parallel, 478
 - parallel_sort_mwms
 - __gnu_parallel, 479
 - parallel_sort_mwms_pu
 - __gnu_parallel, 479
 - parallel_tag
 - __gnu_parallel::parallel_tag, 1307
 - param
 - std::bernoulli_distribution, 2364, 2365
 - std::binomial_distribution, 2376, 2377
 - std::cauchy_distribution, 2396, 2397
 - std::chi_squared_distribution, 2405
 - std::discrete_distribution, 2596
 - std::exponential_distribution, 2612
 - std::extreme_value_distribution, 2617
 - std::fisher_f_distribution, 2621
 - std::gamma_distribution, 2678
 - std::geometric_distribution, 2683, 2684
 - std::lognormal_distribution, 2884, 2885
 - std::negative_binomial_distribution, 3030
 - std::normal_distribution, 3037
 - std::piecewise_constant_distribution, 3148, 3149
 - std::piecewise_linear_distribution, 3155
 - std::poisson_distribution, 3166
 - std::student_t_distribution, 3283
 - std::uniform_int_distribution, 3345
 - std::uniform_real_distribution, 3349
 - std::weibull_distribution, 3402
 - partial_sort
 - sorting_algorithms, 186, 187
 - partial_sort_copy
 - sorting_algorithms, 188
 - partial_sort_minimal_n
 - __gnu_parallel::_Settings, 1282
 - partial_sum
 - std, 726
 - partial_sum.h, 3653
 - partial_sum_dilation
 - __gnu_parallel::_Settings, 1282
 - partial_sum_minimal_n
 - __gnu_parallel::_Settings, 1282
 - partition
 - mutating_algorithms, 138
 - partition.h, 3653
 - _GLIBCXX_VOLATILE, 3654
 - partition_chunk_share
 - __gnu_parallel::_Settings, 1283
 - partition_chunk_size
 - __gnu_parallel::_Settings, 1283
 - partition_copy
 - mutating_algorithms, 139
 - partition_minimal_n
 - __gnu_parallel::_Settings, 1283
 - partition_point
 - mutating_algorithms, 139
 - pbackfail
 - __gnu_cxx::enc_filebuf, 938
 - __gnu_cxx::stdio_filebuf, 1012
 - __gnu_cxx::stdio_sync_filebuf, 1043
 - std::basic_filebuf, 1628
 - std::basic_streambuf, 2179
 - std::basic_stringbuf, 2270
 - pbase
 - __gnu_cxx::enc_filebuf, 939
 - __gnu_cxx::stdio_filebuf, 1012
-

- __gnu_cxx::stdio_sync_filebuf, 1044
 - std::basic_filebuf, 1629
 - std::basic_streambuf, 2180
 - std::basic_stringbuf, 2271
- pbump
 - __gnu_cxx::enc_filebuf, 939
 - __gnu_cxx::stdio_filebuf, 1013
 - __gnu_cxx::stdio_sync_filebuf, 1045
 - std::basic_filebuf, 1630
 - std::basic_streambuf, 2180
 - std::basic_stringbuf, 2272
- peek
 - std::basic_fstream, 1698
 - std::basic_ifstream, 1761
 - std::basic_iostream, 1865
 - std::basic_istream, 1924
 - std::basic_istreamstream, 1985
 - std::basic_stringstream, 2337
- piecewise_construct
 - std, 740
- pod_char_traits.h, 3654
- pointer
 - std::back_insert_iterator, 1606
 - std::front_insert_iterator, 2654
 - std::insert_iterator, 2734
 - std::istream_iterator, 2810
 - std::istreambuf_iterator, 2814
 - std::iterator, 2818
 - std::ostream_iterator, 3129
 - std::ostreambuf_iterator, 3134
 - std::raw_storage_iterator, 3191
 - std::reverse_iterator, 3219
 - std::set, 3233
- Pointer Abstractions, 68
- pointer.h, 3655
- pointer_abstractions
 - allocate_shared, 71
 - get_deleter, 72
 - make_shared, 72
 - operator<<, 72
- pointer_adaptors
 - ptr_fun, 275
- Poisson Distributions, 322
- polar
 - complex_numbers, 52
- Policy-Based Data Structures, 293
- pool_allocator.h, 3658
- pop
 - std::priority_queue, 3171
 - std::queue, 3178
 - std::stack, 3280
- pop_back
 - __gnu_parallel::_-RestrictedBoundedConcurrentQueue, 1274
 - std::deque, 2582
 - std::list, 2852
 - std::vector, 3391
- pop_front
 - __gnu_parallel::_-RestrictedBoundedConcurrentQueue, 1274
 - std::deque, 2583
 - std::forward_list, 2643
 - std::list, 2852
- pop_heap
 - heap_algorithms, 281, 282
- pos_format
 - std::moneypunct, 2969
 - std::moneypunct_byname, 2981
- pos_type
 - __gnu_cxx::enc_filebuf, 932
 - __gnu_cxx::stdio_filebuf, 1001
 - __gnu_cxx::stdio_sync_filebuf, 1038
 - std::basic_filebuf, 1619
 - std::basic_fstream, 1662, 1663
 - std::basic_ifstream, 1734
 - std::basic_ios, 1791
 - std::basic_iostream, 1831
 - std::basic_istream, 1900
 - std::basic_istreamstream, 1960
 - std::basic_ofstream, 2018
 - std::basic_ostream, 2067
 - std::basic_ostreamstream, 2117
 - std::basic_streambuf, 2173
 - std::basic_stringbuf, 2264
 - std::basic_stringstream, 2302, 2303
- position
 - std::match_results, 2922

- positive_sign
 - std::moneypunct, 2970
 - std::moneypunct_byname, 2982
- postypes.h, 3659
- pow
 - complex_numbers, 52, 53
- power
 - SGIextensions, 21
- pptr
 - __gnu_cxx::enc_filebuf, 939
 - __gnu_cxx::stdio_filebuf, 1013
 - __gnu_cxx::stdio_sync_filebuf, 1045
 - std::basic_filebuf, 1630
 - std::basic_streambuf, 2181
 - std::basic_stringbuf, 2272
- precision
 - std::basic_fstream, 1699
 - std::basic_ifstream, 1761
 - std::basic_ios, 1802
 - std::basic_iostream, 1865
 - std::basic_istream, 1925
 - std::basic_istream, 1985, 1986
 - std::basic_ofstream, 2039
 - std::basic_ostream, 2086
 - std::basic_ostringstream, 2136
 - std::basic_stringstream, 2337
 - std::ios_base, 2747, 2748
- prefix
 - std::match_results, 2923
- prev_permutation
 - sorting_algorithms, 189, 190
- priority_queue
 - std::priority_queue, 3170, 3171
- priority_queue.hpp, 3660
- priority_queue_base_dispatch.hpp, 3660
- probabilities
 - std::discrete_distribution, 2596
- profiler.h, 3660
- profiler_algos.h, 3663
- profiler_container_size.h, 3664
- profiler_hash_func.h, 3665
- profiler_hashtable_size.h, 3666
- profiler_list_to_slist.h, 3666
- profiler_list_to_vector.h, 3667
- profiler_map_to_unordered_map.h, 3668
- profiler_node.h, 3669
- profiler_state.h, 3670
- profiler_trace.h, 3671
- profiler_vector_size.h, 3673
- profiler_vector_to_list.h, 3674
- ptr_fun
 - pointer_adaptors, 275
- pubimbue
 - __gnu_cxx::enc_filebuf, 940
 - __gnu_cxx::stdio_filebuf, 1014
 - __gnu_cxx::stdio_sync_filebuf, 1046
 - std::basic_filebuf, 1631
 - std::basic_streambuf, 2181
 - std::basic_stringbuf, 2273
- pubseekoff
 - __gnu_cxx::enc_filebuf, 940
 - __gnu_cxx::stdio_filebuf, 1014
 - __gnu_cxx::stdio_sync_filebuf, 1046
 - std::basic_filebuf, 1631
 - std::basic_streambuf, 2182
 - std::basic_stringbuf, 2273
- pubseekpos
 - __gnu_cxx::enc_filebuf, 941
 - __gnu_cxx::stdio_filebuf, 1015
 - __gnu_cxx::stdio_sync_filebuf, 1046
 - std::basic_filebuf, 1632
 - std::basic_streambuf, 2182
 - std::basic_stringbuf, 2273
- pubsetbuf
 - __gnu_cxx::enc_filebuf, 941
 - __gnu_cxx::stdio_filebuf, 1015
 - __gnu_cxx::stdio_sync_filebuf, 1047
 - std::basic_filebuf, 1632
 - std::basic_streambuf, 2183
 - std::basic_stringbuf, 2274
- pubsync
 - __gnu_cxx::enc_filebuf, 941
 - __gnu_cxx::stdio_filebuf, 1016
 - __gnu_cxx::stdio_sync_filebuf, 1047
 - std::basic_filebuf, 1632
 - std::basic_streambuf, 2183

-
- std::basic_stringbuf, 2274
 - push
 - std::priority_queue, 3172
 - std::queue, 3179
 - std::stack, 3280
 - push_back
 - __gnu_cxx::__versa_string, 887
 - __gnu_debug::basic_string, 1148
 - std::basic_string, 2245
 - std::deque, 2583
 - std::list, 2852
 - std::vector, 3391
 - push_front
 - __gnu_parallel::_-
 - RestrictedBoundedConcurrentQueue, 1274
 - std::deque, 2584
 - std::forward_list, 2644
 - std::list, 2853
 - push_heap
 - heap_algorithms, 282
 - put
 - std::basic_fstream, 1699
 - std::basic_istream, 1866
 - std::basic_ofstream, 2040
 - std::basic_ostream, 2087
 - std::basic_ostringstream, 2137
 - std::basic_stringstream, 2338
 - std::money_put, 2956, 2957
 - std::num_put, 3071–3077
 - std::time_put, 3323, 3324
 - std::time_put_byname, 3328
 - put_money
 - std, 727
 - putback
 - std::basic_fstream, 1700
 - std::basic_ifstream, 1762
 - std::basic_istream, 1866
 - std::basic_istream, 1925
 - std::basic_istream, 1986
 - std::basic_stringstream, 2338
 - pwdword
 - std::basic_fstream, 1701
 - std::basic_ifstream, 1762
 - std::basic_ios, 1803
 - std::basic_istream, 1867
 - std::basic_istream, 1926
 - std::basic_istream, 1987
 - std::basic_ofstream, 2040
 - std::basic_ostream, 2087
 - std::basic_ostringstream, 2137
 - std::basic_stringstream, 2339
 - std::ios_base, 2748
 - qsb_steals
 - __gnu_parallel::_Settings, 1283
 - queue, 3675
 - std::queue, 3177
 - queue.h, 3675
 - _GLIBCXX_VOLATILE, 3676
 - quicksort.h, 3676
 - quiet_NaN
 - std::numeric_limits, 3081
 - radix
 - std::__numeric_limits_base, 1499
 - std::numeric_limits, 3085
 - random, 3677
 - generate_canonical, 229
 - Random Number Distributions, 308
 - Random Number Generation, 228
 - Random Number Generators, 301
 - Random Number Utilities, 330
 - random.h, 3677
 - random.tcc, 3684
 - random_distributions_bernoulli
 - operator<<, 319
 - operator>>, 320, 321
 - operator==, 320
 - random_distributions_normal
 - operator<<, 315
 - operator>>, 316
 - operator==, 316
 - random_distributions_poisson
 - operator<<, 325, 326
 - operator>>, 328, 329
 - operator==, 327, 328
 - random_distributions_uniform
 - operator<<, 310
 - operator>>, 311, 312
 - operator==, 311
 - random_generators
-

- minstd_rand, 303
- minstd_rand0, 303
- mt19937, 303
- mt19937_64, 304
- operator<<, 307
- random_number.h, 3689
- random_sample
 - SGIextensions, 22
- random_sample_n
 - SGIextensions, 22, 23
- random_shuffle
 - mutating_algorithms, 140
- random_shuffle.h, 3690
- random_shuffle_minimal_n
 - __gnu_parallel::Settings, 1283
- range_access.h, 3691
- ratio, 3692
- Rational Arithmetic, 77
- rb_tree, 3692
- rbegin
 - __gnu_cxx::__versa_string, 888
 - __gnu_debug::basic_string, 1149
 - std::basic_string, 2245
 - std::deque, 2584
 - std::list, 2853
 - std::map, 2907
 - std::multimap, 3002
 - std::multiset, 3022
 - std::set, 3248
 - std::vector, 3391, 3392
- rc_string_base.h, 3693
- rdbuf
 - std::basic_fstream, 1701, 1702
 - std::basic_ifstream, 1763
 - std::basic_ios, 1803, 1804
 - std::basic_iostream, 1868
 - std::basic_istream, 1927
 - std::basic_istreamstream, 1987, 1988
 - std::basic_ofstream, 2041
 - std::basic_ostream, 2088
 - std::basic_ostreamstream, 2138
 - std::basic_stringstream, 2339, 2340
- rdstate
 - std::basic_fstream, 1702
 - std::basic_ifstream, 1764
 - std::basic_ios, 1804
 - std::basic_iostream, 1869
 - std::basic_istream, 1928
 - std::basic_istreamstream, 1988
 - std::basic_ofstream, 2042
 - std::basic_ostream, 2089
 - std::basic_ostreamstream, 2139
 - std::basic_stringstream, 2340
- read
 - std::basic_fstream, 1703
 - std::basic_ifstream, 1764
 - std::basic_iostream, 1869
 - std::basic_istream, 1928
 - std::basic_istreamstream, 1989
 - std::basic_stringstream, 2341
- readsome
 - std::basic_fstream, 1703
 - std::basic_ifstream, 1765
 - std::basic_iostream, 1870
 - std::basic_istream, 1929
 - std::basic_istreamstream, 1989
 - std::basic_stringstream, 2342
- ref
 - std, 727
- reference
 - std::back_insert_iterator, 1606
 - std::front_insert_iterator, 2654
 - std::insert_iterator, 2735
 - std::istream_iterator, 2810
 - std::istreambuf_iterator, 2814
 - std::iterator, 2818
 - std::ostream_iterator, 3129
 - std::ostreambuf_iterator, 3134
 - std::raw_storage_iterator, 3191
 - std::reverse_iterator, 3219
 - std::set, 3233
- regex, 3693
 - cregex_token_iterator, 235
 - csub_match, 235
 - isctype, 237
 - operator<, 241–243
 - operator<<, 244
 - operator<=, 244–247
 - operator>, 250–253
 - operator>=, 253–256
 - operator==, 247–250
 - regex, 236

- regex_match, 256–260
- regex_replace, 261
- regex_search, 262–266
- sregex_token_iterator, 236
- ssub_match, 236
- swap, 266, 267
- value, 267
- wcregex_token_iterator, 236
- wcsub_match, 236
- wregex, 236
- wsregex_token_iterator, 237
- wssub_match, 237
- regex.h, 3693
- regex_compiler.h, 3700
- regex_constants.h, 3700
- regex_cursor.h, 3701
- regex_error
 - std::regex_error, 3196
- regex_error.h, 3702
- regex_grep_matcher.h, 3703
- regex_grep_matcher.tcc, 3704
- regex_iterator
 - std::regex_iterator, 3198
- regex_match
 - regex, 256–260
- regex_nfa.h, 3704
- regex_nfa.tcc, 3705
- regex_replace
 - regex, 261
- regex_search
 - regex, 262–266
- regex_token_iterator
 - std::regex_token_iterator, 3202–3204
- regex_traits
 - std::regex_traits, 3208
- register_callback
 - std::basic_fstream, 1704
 - std::basic_ifstream, 1766
 - std::basic_ios, 1805
 - std::basic_iostream, 1871
 - std::basic_istream, 1930
 - std::basic_istream, 1990
 - std::basic_ofstream, 2042
 - std::basic_ostream, 2090
 - std::basic_ostringstream, 2139
 - std::basic_stringstream, 2342
 - std::ios_base, 2749
- Regular Expressions, 229
- release
 - std::auto_ptr, 1602
- remove
 - mutating_algorithms, 141
 - std::forward_list, 2644
 - std::list, 2854
- remove_copy
 - mutating_algorithms, 142
- remove_copy_if
 - mutating_algorithms, 142
- remove_if
 - mutating_algorithms, 143
 - std::forward_list, 2644
 - std::list, 2854
- rend
 - __gnu_cxx::__versa_string, 888, 889
 - __gnu_debug::basic_string, 1149
 - std::basic_string, 2245, 2246
 - std::deque, 2584, 2585
 - std::list, 2854, 2855
 - std::map, 2907
 - std::multimap, 3002
 - std::multiset, 3022
 - std::set, 3248
 - std::vector, 3392
- replace
 - __gnu_cxx::__versa_string, 889–896
 - __gnu_debug::basic_string, 1149–1155
 - mutating_algorithms, 143
 - std::basic_string, 2246–2252
- replace_copy
 - std, 727
- replace_copy_if
 - mutating_algorithms, 144
- replace_if
 - mutating_algorithms, 144
- replace_minimal_n
 - __gnu_parallel::_Settings, 1284
- requested_size
 - __gnu_cxx::temporary_buffer, 1065

-
- std::_Temporary_buffer, 1564
 - reserve
 - __gnu_cxx::__versa_string, 897
 - __gnu_debug::basic_string, 1156
 - std::basic_string, 2253
 - std::vector, 3392
 - reset
 - std::auto_ptr, 1603
 - std::bernoulli_distribution, 2365
 - std::binomial_distribution, 2377
 - std::bitset, 2391
 - std::cauchy_distribution, 2397
 - std::chi_squared_distribution, 2406
 - std::discrete_distribution, 2597
 - std::exponential_distribution, 2612
 - std::extreme_value_distribution, 2618
 - std::fisher_f_distribution, 2621
 - std::gamma_distribution, 2679
 - std::geometric_distribution, 2684
 - std::lognormal_distribution, 2885
 - std::negative_binomial_distribution, 3031
 - std::normal_distribution, 3038
 - std::piecewise_constant_distribution, 3149
 - std::piecewise_linear_distribution, 3155
 - std::poisson_distribution, 3166
 - std::student_t_distribution, 3284
 - std::uniform_int_distribution, 3346
 - std::uniform_real_distribution, 3349
 - std::weibull_distribution, 3403
 - resetiosflags
 - std, 728
 - resize
 - __gnu_cxx::__versa_string, 898
 - __gnu_debug::basic_string, 1157
 - numeric_arrays, 114
 - std::basic_string, 2254
 - std::deque, 2585
 - std::forward_list, 2645
 - std::list, 2855
 - std::vector, 3393
 - result_type
 - __gnu_cxx::__detail::_Ffit_finder, 821
 - __gnu_cxx::binary_compose, 917
 - __gnu_cxx::project1st, 975
 - __gnu_cxx::project2nd, 976
 - __gnu_cxx::select1st, 991
 - __gnu_cxx::select2nd, 992
 - __gnu_cxx::subtractive_rng, 1062
 - __gnu_cxx::unary_compose, 1076
 - __gnu_parallel::EqualFromLess, 1214
 - __gnu_parallel::EqualTo, 1216
 - __gnu_parallel::Less, 1226
 - __gnu_parallel::Lexicographic, 1228
 - __gnu_parallel::_-LexicographicReverse, 1230
 - __gnu_parallel::_Multiplies, 1258
 - __gnu_parallel::_Plus, 1262
 - __gnu_parallel::__binder1st, 1168
 - __gnu_parallel::__binder2nd, 1170
 - __gnu_parallel::__unary_negate, 1207
 - std::_Maybe_unary_or_binary_function< _Res, _T1 >, 1550
 - std::_Maybe_unary_or_binary_function< _Res, _T1, _T2 >, 1551
 - std::bernoulli_distribution, 2363
 - std::binary_function, 2368
 - std::binary_negate, 2370
 - std::binder1st, 2372
 - std::binder2nd, 2373
 - std::binomial_distribution, 2375
 - std::cauchy_distribution, 2396
 - std::chi_squared_distribution, 2404
 - std::const_mem_fun1_ref_t, 2461
 - std::const_mem_fun1_t, 2463
 - std::const_mem_fun_ref_t, 2465
 - std::const_mem_fun_t, 2467
 - std::discard_block_engine, 2588
 - std::discrete_distribution, 2595
 - std::divides, 2600
 - std::equal_to, 2605
 - std::exponential_distribution, 2610
-

-
- std::extreme_value_distribution, 2616
 - std::fisher_f_distribution, 2620
 - std::gamma_distribution, 2676
 - std::geometric_distribution, 2682
 - std::greater, 2686
 - std::greater_equal, 2688
 - std::hash< __gnu_cxx::throw_-value_limit >, 2706
 - std::hash< __gnu_cxx::throw_-value_random >, 2708
 - std::hash< __shared_ptr< _Tp, _Lp > >, 2711
 - std::hash< shared_ptr< _Tp > >, 2714
 - std::hash< unique_ptr< _Tp, _Dp > >, 2719
 - std::independent_bits_engine, 2722
 - std::less, 2823
 - std::less_equal, 2825
 - std::linear_congruential_engine, 2827
 - std::logical_and, 2878
 - std::logical_not, 2880
 - std::logical_or, 2882
 - std::lognormal_distribution, 2884
 - std::mem_fun1_ref_t, 2926
 - std::mem_fun1_t, 2928
 - std::mem_fun_ref_t, 2930
 - std::mem_fun_t, 2932
 - std::minus, 2942
 - std::modulus, 2944
 - std::multiplies, 3006
 - std::negate, 3027
 - std::negative_binomial_distribution, 3029
 - std::normal_distribution, 3035
 - std::not_equal_to, 3042
 - std::owner_less< shared_ptr< _Tp > >, 3140
 - std::owner_less< weak_ptr< _Tp > >, 3141
 - std::piecewise_constant_distribution, 3147
 - std::piecewise_linear_distribution, 3153
 - std::plus, 3159
 - std::pointer_to_binary_function, 3161
 - std::pointer_to_unary_function, 3163
 - std::poisson_distribution, 3164
 - std::random_device, 3181
 - std::seed_seq, 3227
 - std::shuffle_order_engine, 3268
 - std::student_t_distribution, 3282
 - std::unary_function, 3339
 - std::unary_negate, 3341
 - std::uniform_int_distribution, 3344
 - std::uniform_real_distribution, 3348
 - std::weibull_distribution, 3401
 - rethrow_exception
 - exceptions, 35
 - rethrow_if_nested
 - exceptions, 35
 - return_temporary_buffer
 - std, 728
 - reverse
 - mutating_algorithms, 145
 - std::forward_list, 2646
 - std::list, 2856
 - reverse_copy
 - mutating_algorithms, 145
 - reverse_iterator
 - std::reverse_iterator, 3220
 - std::set, 3233
 - rfind
 - __gnu_cxx::__versa_string, 899, 900
 - __gnu_debug::basic_string, 1157–1159
 - std::basic_string, 2255, 2256
 - riemann_zeta
 - tr1_math_spec_func, 124
 - right
 - std, 729
 - std::basic_fstream, 1721
 - std::basic_ifstream, 1781
 - std::basic_ios, 1816
 - std::basic_iostream, 1888
 - std::basic_istream, 1945
 - std::basic_istream, 2006
-

- std::basic_ofstream, 2056
- std::basic_ostream, 2103
- std::basic_ostringstream, 2154
- std::basic_stringstream, 2360
- std::ios_base, 2758
- rope, 3705
- ropeimpl.h, 3709
- rotate
 - mutating_algorithms, 146
- rotate_copy
 - mutating_algorithms, 147
- round_to_nearest
 - std, 641
- round_toward_infinity
 - std, 641
- round_toward_neg_infinity
 - std, 641
- round_toward_zero
 - std, 641
- round_error
 - std::numeric_limits, 3081
- round_style
 - std::__numeric_limits_base, 1499
 - std::numeric_limits, 3085
- runtime_error
 - std::runtime_error, 3226
- safe_base.h, 3710
- safe_iterator.h, 3711
- safe_iterator.tcc, 3712
- safe_sequence.h, 3713
- safe_sequence.tcc, 3713
- sbumpc
 - __gnu_cxx::enc_filebuf, 942
 - __gnu_cxx::stdio_filebuf, 1016
 - __gnu_cxx::stdio_sync_filebuf, 1048
 - std::basic_filebuf, 1633
 - std::basic_streambuf, 2183
 - std::basic_stringbuf, 2275
- scan_is
 - std::__ctype_abstract_base, 1402
 - std::ctype, 2479
 - std::ctype< char >, 2494
 - std::ctype< wchar_t >, 2512
 - std::ctype_byname, 2529
 - std::ctype_byname< char >, 2543
- scan_not
 - std::__ctype_abstract_base, 1402
 - std::ctype, 2480
 - std::ctype< char >, 2494
 - std::ctype< wchar_t >, 2512
 - std::ctype_byname, 2530
 - std::ctype_byname< char >, 2543
- scientific
 - std, 729
 - std::basic_fstream, 1722
 - std::basic_ifstream, 1781
 - std::basic_ios, 1817
 - std::basic_iostream, 1888
 - std::basic_istream, 1945
 - std::basic_istreamstream, 2006
 - std::basic_ofstream, 2056
 - std::basic_ostream, 2104
 - std::basic_ostringstream, 2154
 - std::basic_stringstream, 2361
 - std::ios_base, 2758
- search
 - non_mutating_algorithms, 167
- search.h, 3713
- search_minimal_n
 - __gnu_parallel::_Settings, 1284
- search_n
 - non_mutating_algorithms, 168, 169
- second
 - __gnu_parallel::_IteratorPair, 1221
 - std::pair, 3145
 - std::sub_match, 3291
- second_argument_type
 - __gnu_cxx::project1st, 975
 - __gnu_cxx::project2nd, 977
 - __gnu_parallel::_EqualFromLess, 1214
 - __gnu_parallel::_EqualTo, 1216
 - __gnu_parallel::_Less, 1226
 - __gnu_parallel::_Lexicographic, 1228
 - __gnu_parallel::_LexicographicReverse, 1230
 - __gnu_parallel::_Multiplies, 1258
 - __gnu_parallel::_Plus, 1262

-
- std::_Maybe_unary_or_binary_-
function< _Res, _T1, _T2 >,
1551
 - std::binary_function, 2368
 - std::binary_negate, 2370
 - std::const_mem_fun1_ref_t, 2461
 - std::const_mem_fun1_t, 2463
 - std::divides, 2600
 - std::equal_to, 2605
 - std::greater, 2686
 - std::greater_equal, 2688
 - std::less, 2823
 - std::less_equal, 2825
 - std::logical_and, 2878
 - std::logical_or, 2882
 - std::mem_fun1_ref_t, 2926
 - std::mem_fun1_t, 2928
 - std::minus, 2942
 - std::modulus, 2944
 - std::multiplies, 3006
 - std::not_equal_to, 3042
 - std::owner_less< shared_ptr< _Tp
> >, 3140
 - std::owner_less< weak_ptr< _Tp >
>, 3141
 - std::plus, 3159
 - std::pointer_to_binary_function,
3161
 - second_type
 __gnu_parallel::_IteratorPair, 1221
 - std::pair, 3144
 - std::sub_match, 3288
 - seconds
 std::chrono, 786
 - seed
 std::discard_block_engine, 2591,
 2592
 - std::independent_bits_engine, 2725,
 2726
 - std::linear_congruential_engine,
 2829
 - std::shuffle_order_engine, 3271,
 3272
 - seed_seq
 std::seed_seq, 3227
 - seekdir
 - std::basic_fstream, 1663
 - std::basic_ifstream, 1734
 - std::basic_ios, 1791
 - std::basic_iostream, 1832
 - std::basic_istream, 1900
 - std::basic_istreamstream, 1960
 - std::basic_ofstream, 2018
 - std::basic_ostream, 2068
 - std::basic_ostreamstream, 2117
 - std::basic_stringstream, 2303
 - std::ios_base, 2744
 - seekg
 std::basic_fstream, 1704, 1705
 - std::basic_ifstream, 1766, 1767
 - std::basic_iostream, 1871, 1872
 - std::basic_istream, 1930, 1931
 - std::basic_istreamstream, 1990, 1991
 - std::basic_stringstream, 2343
 - seekoff
 __gnu_cxx::enc_filebuf, 942
 - __gnu_cxx::stdio_filebuf, 1016
 - __gnu_cxx::stdio_sync_filebuf,
 1048
 - std::basic_filebuf, 1633
 - std::basic_streambuf, 2184
 - std::basic_stringbuf, 2275
 - seekp
 std::basic_fstream, 1706
 - std::basic_iostream, 1873
 - std::basic_ofstream, 2043
 - std::basic_ostream, 2090, 2091
 - std::basic_ostreamstream, 2140
 - std::basic_stringstream, 2344, 2345
 - seekpos
 __gnu_cxx::enc_filebuf, 942
 - __gnu_cxx::stdio_filebuf, 1017
 - __gnu_cxx::stdio_sync_filebuf,
 1048
 - std::basic_filebuf, 1634
 - std::basic_streambuf, 2184
 - std::basic_stringbuf, 2276
 - sentry
 std::basic_istream::sentry, 1948
 - std::basic_ostream::sentry, 2106
 - Sequences, 25
 - sequential
-

-
- __gnu_parallel, 431
 - set, 3714
 - __gnu_parallel::_Settings, 1278
 - std::bitset, 2392
 - std::set, 3234–3236
 - Set Operation, 193
 - set.h, 3715, 3716
 - set_algorithms
 - includes, 194, 195
 - set_difference, 195, 196
 - set_intersection, 196, 197
 - set_symmetric_difference, 198
 - set_union, 199, 200
 - set_difference
 - set_algorithms, 195, 196
 - set_difference_minimal_n
 - __gnu_parallel::_Settings, 1284
 - set_intersection
 - set_algorithms, 196, 197
 - set_intersection_minimal_n
 - __gnu_parallel::_Settings, 1284
 - set_new_handler
 - std, 729
 - set_num_threads
 - __gnu_parallel::balanced_-
quicksort_tag, 1289
 - __gnu_parallel::balanced_tag, 1290
 - __gnu_parallel::default_parallel_-
tag, 1293
 - __gnu_parallel::exact_tag, 1295
 - __gnu_parallel::multiway_-
mergesort_exact_tag, 1298
 - __gnu_parallel::multiway_-
mergesort_sampling_tag,
1300
 - __gnu_parallel::multiway_-
mergesort_tag, 1301
 - __gnu_parallel::omp_loop_static_-
tag, 1303
 - __gnu_parallel::omp_loop_tag,
1304
 - __gnu_parallel::parallel_tag, 1307
 - __gnu_parallel::quicksort_tag, 1309
 - __gnu_parallel::sampling_tag, 1310
 - __gnu_parallel::unbalanced_tag,
1312
 - set_operations.h, 3717
 - set_symmetric_difference
 - set_algorithms, 198
 - set_symmetric_difference_minimal_n
 - __gnu_parallel::_Settings, 1284
 - set_terminate
 - exceptions, 35
 - set_unexpected
 - exceptions, 35
 - set_union
 - set_algorithms, 199, 200
 - set_union_minimal_n
 - __gnu_parallel::_Settings, 1285
 - setbase
 - std, 729
 - setbuf
 - __gnu_cxx::enc_filebuf, 943
 - __gnu_cxx::stdio_filebuf, 1017
 - __gnu_cxx::stdio_sync_filebuf,
1049
 - std::basic_filebuf, 1634
 - std::basic_streambuf, 2185
 - std::basic_stringbuf, 2276
 - setf
 - std::basic_fstream, 1707
 - std::basic_ifstream, 1768
 - std::basic_ios, 1805, 1806
 - std::basic_iostream, 1874
 - std::basic_istream, 1932
 - std::basic_istreamstream, 1992
 - std::basic_ofstream, 2044
 - std::basic_ostream, 2091
 - std::basic_ostringstream, 2141
 - std::basic_stringstream, 2345, 2346
 - std::ios_base, 2749
 - setfill
 - std, 730
 - setg
 - __gnu_cxx::enc_filebuf, 943
 - __gnu_cxx::stdio_filebuf, 1018
 - __gnu_cxx::stdio_sync_filebuf,
1049
 - std::basic_filebuf, 1635
 - std::basic_streambuf, 2185
 - std::basic_stringbuf, 2277
 - setiosflags
-

-
- std, 730
 - setp
 - __gnu_cxx::enc_filebuf, 944
 - __gnu_cxx::stdio_filebuf, 1019
 - __gnu_cxx::stdio_sync_filebuf, 1050
 - std::basic_filebuf, 1635
 - std::basic_streambuf, 2186
 - std::basic_stringbuf, 2277
 - setprecision
 - std, 730
 - setstate
 - std::basic_fstream, 1708
 - std::basic_ifstream, 1768
 - std::basic_ios, 1806
 - std::basic_iostream, 1874
 - std::basic_istream, 1932
 - std::basic_istream, 1993
 - std::basic_ofstream, 2045
 - std::basic_ostream, 2092
 - std::basic_ostringstream, 2142
 - std::basic_stringstream, 2346
 - settings.h, 3718
 - _GLIBCXX_PARALLEL_CONDITION, 3719
 - setw
 - std, 731
 - sgetc
 - __gnu_cxx::enc_filebuf, 944
 - __gnu_cxx::stdio_filebuf, 1019
 - __gnu_cxx::stdio_sync_filebuf, 1050
 - std::basic_filebuf, 1636
 - std::basic_streambuf, 2186
 - std::basic_stringbuf, 2278
 - sgetn
 - __gnu_cxx::enc_filebuf, 944
 - __gnu_cxx::stdio_filebuf, 1019
 - __gnu_cxx::stdio_sync_filebuf, 1051
 - std::basic_filebuf, 1636
 - std::basic_streambuf, 2187
 - std::basic_stringbuf, 2278
 - SGL, 10
 - SGLextensions
 - _Find_first, 15
 - _Find_next, 15
 - _Unchecked_flip, 16
 - _Unchecked_reset, 16
 - _Unchecked_set, 16
 - _Unchecked_test, 16
 - __median, 14
 - compose1, 17
 - compose2, 17
 - constant0, 17
 - constant1, 17
 - constant2, 17
 - copy_n, 18
 - distance, 18
 - identity_element, 19
 - iota, 19
 - is_heap, 19, 20
 - is_sorted, 20
 - lexicographical_compare_3way, 20
 - power, 21
 - random_sample, 22
 - random_sample_n, 22, 23
 - uninitialized_copy_n, 23
 - shared_future
 - std::shared_future, 3251, 3252
 - std::shared_future< _Res & >, 3254
 - std::shared_future< void >, 3257
 - shared_ptr
 - std::shared_ptr, 3261–3266
 - shared_ptr.h, 3719
 - shared_ptr_base.h, 3721
 - shift
 - numeric_arrays, 114
 - showbase
 - std, 731
 - std::basic_fstream, 1722
 - std::basic_ifstream, 1781
 - std::basic_ios, 1817
 - std::basic_iostream, 1888
 - std::basic_istream, 1945
 - std::basic_istream, 2006
 - std::basic_ofstream, 2056
 - std::basic_ostream, 2104
 - std::basic_ostringstream, 2154
 - std::basic_stringstream, 2361
 - std::ios_base, 2758
 - showmanyc
-

- `__gnu_cxx::enc_filebuf`, 945
 - `__gnu_cxx::stdio_filebuf`, 1020
 - `__gnu_cxx::stdio_sync_filebuf`, 1051
 - `std::basic_filebuf`, 1637
 - `std::basic_streambuf`, 2187
 - `std::basic_stringbuf`, 2279
- showpoint
 - `std`, 731
 - `std::basic_fstream`, 1722
 - `std::basic_ifstream`, 1781
 - `std::basic_ios`, 1817
 - `std::basic_iostream`, 1888
 - `std::basic_istream`, 1945
 - `std::basic_istreamstream`, 2006
 - `std::basic_ofstream`, 2057
 - `std::basic_ostream`, 2104
 - `std::basic_ostreamstream`, 2154
 - `std::basic_stringstream`, 2361
 - `std::ios_base`, 2759
- showpos
 - `std`, 731
 - `std::basic_fstream`, 1722
 - `std::basic_ifstream`, 1782
 - `std::basic_ios`, 1817
 - `std::basic_iostream`, 1889
 - `std::basic_istream`, 1946
 - `std::basic_istreamstream`, 2007
 - `std::basic_ofstream`, 2057
 - `std::basic_ostream`, 2104
 - `std::basic_ostreamstream`, 2154
 - `std::basic_stringstream`, 2361
 - `std::ios_base`, 2759
- shrink_to_fit
 - `__gnu_cxx::__versa_string`, 901
 - `__gnu_debug::basic_string`, 1159
 - `std::basic_string`, 2257
 - `std::deque`, 2586
 - `std::vector`, 3394
- shuffle
 - mutating_algorithms, 147
- shuffle_order_engine
 - `std::shuffle_order_engine`, 3269, 3270
- signaling_NaN
 - `std::numeric_limits`, 3082
- sin
 - complex_numbers, 53
- sinh
 - complex_numbers, 53
- size
 - `__gnu_cxx::__versa_string`, 901
 - `__gnu_cxx::temporary_buffer`, 1065
 - `__gnu_debug::basic_string`, 1159
 - numeric_arrays, 115
 - `std::_Temporary_buffer`, 1565
 - `std::basic_string`, 2257
 - `std::bitset`, 2392
 - `std::deque`, 2586
 - `std::list`, 2856
 - `std::map`, 2908
 - `std::match_results`, 2923
 - `std::multimap`, 3003
 - `std::multiset`, 3022
 - `std::priority_queue`, 3172
 - `std::queue`, 3179
 - `std::set`, 3248
 - `std::stack`, 3280
 - `std::vector`, 3394
- size_type
 - `std::set`, 3233
- skipws
 - `std`, 731
 - `std::basic_fstream`, 1722
 - `std::basic_ifstream`, 1782
 - `std::basic_ios`, 1817
 - `std::basic_iostream`, 1889
 - `std::basic_istream`, 1946
 - `std::basic_istreamstream`, 2007
 - `std::basic_ofstream`, 2057
 - `std::basic_ostream`, 2104
 - `std::basic_ostreamstream`, 2155
 - `std::basic_stringstream`, 2361
 - `std::ios_base`, 2759
- sleep_for
 - `std::this_thread`, 811
- sleep_until
 - `std::this_thread`, 812
- slice
 - numeric_arrays, 92
- slice_array
 - numeric_arrays, 93

- slice_array.h, 3723
- slist, 3724
- snextc
 - __gnu_cxx::enc_filebuf, 945
 - __gnu_cxx::stdio_filebuf, 1021
 - __gnu_cxx::stdio_sync_filebuf, 1052
 - std::basic_filebuf, 1637
 - std::basic_streambuf, 2188
 - std::basic_stringbuf, 2279
- sort
 - sorting_algorithms, 190, 191
 - std::forward_list, 2646
 - std::list, 2856
- sort.h, 3725
- sort_heap
 - heap_algorithms, 283
- sort_minimal_n
 - __gnu_parallel::_Settings, 1285
- sort_mwms_oversampling
 - __gnu_parallel::_Settings, 1285
- sort_qs_num_samples_preset
 - __gnu_parallel::_Settings, 1285
- sort_qsb_base_case_maximal_n
 - __gnu_parallel::_Settings, 1285
- Sorting, 170
- sorting_algorithms
 - inplace_merge, 173
 - is_sorted, 174
 - is_sorted_until, 175
 - lexicographical_compare, 176
 - max, 177, 178
 - max_element, 178
 - merge, 179, 180
 - min, 180, 181
 - min_element, 181, 182
 - minmax, 182, 183
 - minmax_element, 183, 184
 - next_permutation, 184, 185
 - nth_element, 185, 186
 - partial_sort, 186, 187
 - partial_sort_copy, 188
 - prev_permutation, 189, 190
 - sort, 190, 191
 - stable_sort, 191, 192
- sph_bessel
 - tr1_math_spec_func, 124
- sph_legendre
 - tr1_math_spec_func, 124
- sph_neumann
 - tr1_math_spec_func, 124
- splice
 - std::list, 2857, 2858
- splice_after
 - std::forward_list, 2646, 2647
- sputbackc
 - __gnu_cxx::enc_filebuf, 946
 - __gnu_cxx::stdio_filebuf, 1021
 - __gnu_cxx::stdio_sync_filebuf, 1052
 - std::basic_filebuf, 1638
 - std::basic_streambuf, 2188
 - std::basic_stringbuf, 2280
- sputc
 - __gnu_cxx::enc_filebuf, 946
 - __gnu_cxx::stdio_filebuf, 1021
 - __gnu_cxx::stdio_sync_filebuf, 1053
 - std::basic_filebuf, 1638
 - std::basic_streambuf, 2189
 - std::basic_stringbuf, 2280
- sputn
 - __gnu_cxx::enc_filebuf, 947
 - __gnu_cxx::stdio_filebuf, 1022
 - __gnu_cxx::stdio_sync_filebuf, 1053
 - std::basic_filebuf, 1639
 - std::basic_streambuf, 2189
 - std::basic_stringbuf, 2281
- sqrt
 - complex_numbers, 54
- sregex_token_iterator
 - regex, 236
- sso_string_base.h, 3726
- sstream, 3726
- sstream.tcc, 3727
- ssub_match
 - regex, 236
- stable_partition
 - mutating_algorithms, 148
- stable_sort
 - sorting_algorithms, 191, 192

- stack, 3727
 - std::stack, 3279
- standard_policies.hpp, 3728
- start
 - numeric_arrays, 115
- state
 - std::fpos, 2651, 2652
- static_pointer_cast
 - std, 732
- std, 490
 - _Construct, 656
 - _Destroy, 656
 - _final_insertion_sort, 641
 - _find, 642
 - _find_if, 642
 - _find_if_not, 643
 - _gcd, 643
 - _heap_select, 643, 644
 - _inplace_stable_partition, 644
 - _inplace_stable_sort, 644, 645
 - _insertion_sort, 645
 - _introsort_loop, 645, 646
 - _invoke, 739
 - _joinit, 739
 - _lg, 646
 - _merge_adaptive, 646, 647
 - _merge_backward, 647
 - _merge_without_buffer, 648
 - _move_median_first, 648
 - _partition, 649
 - _reverse, 649, 650
 - _rotate, 650
 - _rotate_adaptive, 651
 - _search_n, 651, 652
 - _stable_partition_adaptive, 652
 - _unguarded_insertion_sort, 652, 653
 - _unguarded_linear_insert, 653
 - _unguarded_partition, 653, 654
 - _unguarded_partition_pivot, 654
 - _unique_copy, 654–656
 - accumulate, 657
 - acos, 658
 - acosh, 658
 - addressof, 658
 - adjacent_difference, 658, 659
 - advance, 660
 - arg, 660
 - asin, 660
 - asinh, 661
 - atan, 661
 - atanh, 661
 - begin, 661, 662
 - bind, 662
 - boolalpha, 663
 - cerr, 739
 - cin, 739
 - clog, 739
 - const_pointer_cast, 663
 - cout, 739
 - cref, 663
 - dec, 663
 - denorm_absent, 641
 - denorm_indeterminate, 641
 - denorm_present, 641
 - distance, 664
 - dynamic_pointer_cast, 664
 - end, 665
 - endl, 666
 - ends, 666
 - fabs, 666
 - fixed, 667
 - float_denorm_style, 640
 - float_round_style, 641
 - flush, 667
 - forward, 667
 - get_money, 667
 - get_temporary_buffer, 668
 - getline, 668–670
 - has_facet, 671
 - hex, 671
 - inner_product, 672
 - internal, 673
 - iota, 673
 - isalnum, 674
 - isalpha, 674
 - isctrl, 674
 - isdigit, 674
 - isgraph, 674
 - islower, 674
 - isprint, 675
 - ispunct, 675

- isspace, 675
- isupper, 675
- isxdigit, 675
- left, 675
- make_pair, 676
- new_handler, 639
- noboolalpha, 676
- noshowbase, 676
- noshowpoint, 677
- noshowpos, 677
- noskipws, 677
- nounitbuf, 677
- nouppercase, 677
- oct, 678
- operator<, 685–691
- operator<=, 691–699
- operator<=, 699–702
- operator>, 710–714
- operator>>, 718–724
- operator>=, 714–717
- operator[^], 725
- operator+, 683, 684
- operator==, 703–710
- operator&, 682
- partial_sum, 726
- piecewise_construct, 740
- put_money, 727
- ref, 727
- replace_copy, 727
- resetiosflags, 728
- return_temporary_buffer, 728
- right, 729
- round_to_nearest, 641
- round_toward_infinity, 641
- round_toward_neg_infinity, 641
- round_toward_zero, 641
- scientific, 729
- set_new_handler, 729
- setbase, 729
- setfill, 730
- setiosflags, 730
- setprecision, 730
- setw, 731
- showbase, 731
- showpoint, 731
- showpos, 731
- skipws, 731
- static_pointer_cast, 732
- streamoff, 639
- streampos, 639
- streamsize, 640
- swap, 732–734
- tolower, 735
- toupper, 735
- u16streampos, 640
- u32streampos, 640
- uninitialized_copy, 735
- uninitialized_copy_n, 736
- uninitialized_fill, 736
- uninitialized_fill_n, 737
- unitbuf, 737
- uppercase, 737
- use_facet, 737
- wcerr, 740
- wcin, 740
- wclog, 740
- wcout, 740
- ws, 738
- wstreampos, 640
- std::__atomic0::__atomic_base, 1372
- std::__atomic0::atomic_address, 1374
- std::__atomic0::atomic_flag, 1376
- std::__atomic2::__atomic_base, 1377
- std::__atomic2::atomic_address, 1379
- std::__atomic2::atomic_flag, 1381
- std::__atomic_flag_base, 1382
- std::__basic_future, 1383
- std::__M_get_result, 1384
- std::__codecvt_abstract_base, 1384
- do_out, 1386
- in, 1387
- out, 1388
- unshift, 1389
- std::__ctype_abstract_base, 1390
- char_type, 1393
- do_is, 1393
- do_narrow, 1394
- do_scan_is, 1395
- do_scan_not, 1396
- do_tolower, 1396, 1397
- do_toupper, 1397, 1398
- do_widen, 1398, 1399

- is, 1400
- narrow, 1401
- scan_is, 1402
- scan_not, 1402
- tolower, 1403
- toupper, 1404
- widen, 1405
- std::__debug, 740
 - operator<=, 747
 - operator>, 747
 - operator>=, 747
 - swap, 747
- std::__debug::bitset, 1406
- std::__debug::deque, 1408
 - _M_attach, 1411
 - _M_attach_single, 1411
 - _M_const_iterators, 1413
 - _M_detach, 1411
 - _M_detach_all, 1411
 - _M_detach_single, 1411
 - _M_detach_singular, 1412
 - _M_get_mutex, 1412
 - _M_invalidate_all, 1412
 - _M_invalidate_if, 1412
 - _M_iterators, 1413
 - _M_revalidate_singular, 1412
 - _M_swap, 1413
 - _M_transfer_from_if, 1413
 - _M_version, 1413
- std::__debug::forward_list, 1414
 - _M_attach, 1417
 - _M_attach_single, 1417
 - _M_const_iterators, 1419
 - _M_detach, 1417
 - _M_detach_all, 1417
 - _M_detach_single, 1418
 - _M_detach_singular, 1418
 - _M_get_mutex, 1418
 - _M_invalidate_all, 1418
 - _M_invalidate_if, 1418
 - _M_iterators, 1419
 - _M_revalidate_singular, 1418
 - _M_swap, 1419
 - _M_transfer_from_if, 1419
 - _M_version, 1420
- std::__debug::list, 1420
 - _M_attach, 1424
 - _M_attach_single, 1424
 - _M_const_iterators, 1426
 - _M_detach, 1424
 - _M_detach_all, 1424
 - _M_detach_single, 1424
 - _M_detach_singular, 1425
 - _M_get_mutex, 1425
 - _M_invalidate_all, 1425
 - _M_invalidate_if, 1425
 - _M_iterators, 1426
 - _M_revalidate_singular, 1425
 - _M_swap, 1426
 - _M_transfer_from_if, 1426
 - _M_version, 1426
- std::__debug::map, 1427
 - _M_attach, 1430
 - _M_attach_single, 1430
 - _M_const_iterators, 1432
 - _M_detach, 1430
 - _M_detach_all, 1430
 - _M_detach_single, 1430
 - _M_detach_singular, 1431
 - _M_get_mutex, 1431
 - _M_invalidate_all, 1431
 - _M_invalidate_if, 1431
 - _M_iterators, 1432
 - _M_revalidate_singular, 1431
 - _M_swap, 1432
 - _M_transfer_from_if, 1432
 - _M_version, 1433
- std::__debug::multimap, 1433
 - _M_attach, 1436
 - _M_attach_single, 1436
 - _M_const_iterators, 1438
 - _M_detach, 1436
 - _M_detach_all, 1436
 - _M_detach_single, 1436
 - _M_detach_singular, 1437
 - _M_get_mutex, 1437
 - _M_invalidate_all, 1437
 - _M_invalidate_if, 1437
 - _M_iterators, 1438
 - _M_revalidate_singular, 1437
 - _M_swap, 1438
 - _M_transfer_from_if, 1438

- [_M_version, 1439](#)
- [std::__debug::multiset, 1439](#)
 - [_M_attach, 1442](#)
 - [_M_attach_single, 1442](#)
 - [_M_const_iterators, 1444](#)
 - [_M_detach, 1442](#)
 - [_M_detach_all, 1442](#)
 - [_M_detach_single, 1442](#)
 - [_M_detach_singular, 1443](#)
 - [_M_get_mutex, 1443](#)
 - [_M_invalidate_all, 1443](#)
 - [_M_invalidate_if, 1443](#)
 - [_M_iterators, 1444](#)
 - [_M_revalidate_singular, 1443](#)
 - [_M_swap, 1444](#)
 - [_M_transfer_from_if, 1444](#)
 - [_M_version, 1445](#)
- [std::__debug::set, 1445](#)
 - [_M_attach, 1449](#)
 - [_M_attach_single, 1449](#)
 - [_M_const_iterators, 1451](#)
 - [_M_detach, 1449](#)
 - [_M_detach_all, 1449](#)
 - [_M_detach_single, 1449](#)
 - [_M_detach_singular, 1449](#)
 - [_M_get_mutex, 1449](#)
 - [_M_invalidate_all, 1450](#)
 - [_M_invalidate_if, 1450](#)
 - [_M_iterators, 1451](#)
 - [_M_revalidate_singular, 1450](#)
 - [_M_swap, 1450](#)
 - [_M_transfer_from_if, 1450](#)
 - [_M_version, 1451](#)
- [std::__debug::unordered_map, 1452](#)
 - [_M_attach, 1454](#)
 - [_M_attach_single, 1454](#)
 - [_M_const_iterators, 1457](#)
 - [_M_detach, 1455](#)
 - [_M_detach_all, 1455](#)
 - [_M_detach_single, 1455](#)
 - [_M_detach_singular, 1455](#)
 - [_M_get_mutex, 1455](#)
 - [_M_invalidate_all, 1455](#)
 - [_M_invalidate_if, 1456](#)
 - [_M_iterators, 1457](#)
 - [_M_revalidate_singular, 1456](#)
 - [_M_swap, 1456](#)
 - [_M_transfer_from_if, 1456](#)
 - [_M_version, 1457](#)
- [std::__debug::unordered_multimap, 1457](#)
 - [_M_attach, 1460](#)
 - [_M_attach_single, 1460](#)
 - [_M_const_iterators, 1462](#)
 - [_M_detach, 1460](#)
 - [_M_detach_all, 1460](#)
 - [_M_detach_single, 1460](#)
 - [_M_detach_singular, 1461](#)
 - [_M_get_mutex, 1461](#)
 - [_M_invalidate_all, 1461](#)
 - [_M_invalidate_if, 1461](#)
 - [_M_iterators, 1462](#)
 - [_M_revalidate_singular, 1461](#)
 - [_M_swap, 1462](#)
 - [_M_transfer_from_if, 1462](#)
 - [_M_version, 1463](#)
- [std::__debug::unordered_multiset, 1463](#)
 - [_M_attach, 1465](#)
 - [_M_attach_single, 1465](#)
 - [_M_const_iterators, 1467](#)
 - [_M_detach, 1465](#)
 - [_M_detach_all, 1466](#)
 - [_M_detach_single, 1466](#)
 - [_M_detach_singular, 1466](#)
 - [_M_get_mutex, 1466](#)
 - [_M_invalidate_all, 1466](#)
 - [_M_invalidate_if, 1466](#)
 - [_M_iterators, 1468](#)
 - [_M_revalidate_singular, 1467](#)
 - [_M_swap, 1467](#)
 - [_M_transfer_from_if, 1467](#)
 - [_M_version, 1468](#)
- [std::__debug::unordered_set, 1468](#)
 - [_M_attach, 1471](#)
 - [_M_attach_single, 1471](#)
 - [_M_const_iterators, 1473](#)
 - [_M_detach, 1471](#)
 - [_M_detach_all, 1471](#)
 - [_M_detach_single, 1472](#)
 - [_M_detach_singular, 1472](#)
 - [_M_get_mutex, 1472](#)
 - [_M_invalidate_all, 1472](#)
 - [_M_invalidate_if, 1472](#)

- [_M_iterators](#), 1473
 - [_M_revalidate_singular](#), 1473
 - [_M_swap](#), 1473
 - [_M_transfer_from_if](#), 1473
 - [_M_version](#), 1474
- [std::__debug::vector](#), 1474
 - [_M_attach](#), 1478
 - [_M_attach_single](#), 1478
 - [_M_const_iterators](#), 1480
 - [_M_detach](#), 1478
 - [_M_detach_all](#), 1478
 - [_M_detach_single](#), 1479
 - [_M_detach_singular](#), 1479
 - [_M_get_mutex](#), 1479
 - [_M_invalidate_all](#), 1479
 - [_M_invalidate_if](#), 1479
 - [_M_iterators](#), 1480
 - [_M_revalidate_singular](#), 1479
 - [_M_swap](#), 1480
 - [_M_transfer_from_if](#), 1480
 - [_M_version](#), 1481
- [vector](#), 1478
- [std::__declval_protector](#), 1481
- [std::__detail](#), 748
- [std::__detail::__List_node_base](#), 1482
- [std::__exception_ptr::exception_ptr](#), 1483
- [std::__future_base](#), 1483
- [std::__future_base::__Ptr](#), 1485
- [std::__future_base::__Result](#), 1486
- [std::__future_base::__Result< _Res & >](#), 1487
- [std::__future_base::__Result< void >](#), 1488
- [std::__future_base::__Result_alloc](#), 1488
- [std::__future_base::__Result_base](#), 1490
- [std::__future_base::__State](#), 1491
- [std::__has_iterator_category_helper](#), 1492
- [std::__is_location_invariant](#), 1492
- [std::__is_member_pointer_helper](#), 1493
- [std::__numeric_limits_base](#), 1494
 - [digits](#), 1496
 - [digits10](#), 1496
 - [has_denorm](#), 1496
 - [has_denorm_loss](#), 1496
 - [has_infinity](#), 1496
 - [has_quiet_NaN](#), 1496
 - [has_signaling_NaN](#), 1496
 - [is_bounded](#), 1497
 - [is_exact](#), 1497
 - [is_iec559](#), 1497
 - [is_integer](#), 1497
 - [is_modulo](#), 1497
 - [is_signed](#), 1497
 - [is_specialized](#), 1498
 - [max_digits10](#), 1498
 - [max_exponent](#), 1498
 - [max_exponent10](#), 1498
 - [min_exponent](#), 1498
 - [min_exponent10](#), 1498
 - [radix](#), 1499
 - [round_style](#), 1499
 - [tinyness_before](#), 1499
 - [traps](#), 1499
- [std::__parallel](#), 749
- [std::__parallel::__CRandNumber](#), 1499
- [std::__profile](#), 774
 - [operator<=](#), 781
 - [operator>](#), 781
 - [operator>=](#), 782
 - [swap](#), 782
- [std::__profile::bitset](#), 1500
- [std::__profile::deque](#), 1502
- [std::__profile::forward_list](#), 1504
- [std::__profile::list](#), 1505
- [std::__profile::map](#), 1508
- [std::__profile::multimap](#), 1510
- [std::__profile::multiset](#), 1512
- [std::__profile::set](#), 1514
- [std::__profile::unordered_map](#), 1516
- [std::__profile::unordered_multimap](#), 1518
- [std::__profile::unordered_multiset](#), 1520
- [std::__profile::unordered_set](#), 1521
- [std::__Base_bitset](#), 1523
 - [_M_w](#), 1524
- [std::__Base_bitset< 0 >](#), 1524
- [std::__Base_bitset< 1 >](#), 1525
- [std::__Build_index_tuple](#), 1527
- [std::__Deque_base](#), 1528
 - [_M_initialize_map](#), 1529
- [std::__Deque_iterator](#), 1530
 - [_M_set_node](#), 1532

-
- std::_Derives_from_binary_function, 1532
 - std::_Derives_from_unary_function, 1533
 - std::_Function_base, 1533
 - std::_Function_to_function_pointer, 1535
 - std::_Fwd_list_base, 1535
 - std::_Fwd_list_const_iterator, 1537
 - std::_Fwd_list_iterator, 1538
 - std::_Fwd_list_node, 1539
 - std::_Fwd_list_node_base, 1541
 - std::_Index_tuple, 1542
 - std::_List_base, 1542
 - std::_List_const_iterator, 1543
 - std::_List_iterator, 1544
 - std::_List_node, 1545
 - std::_M_data, 1547
 - std::_Maybe_get_result_type, 1547
 - std::_Maybe_unary_or_binary_function, 1548
 - std::_Maybe_unary_or_binary_function< _Res, _T1 >, 1548
 - std::_argument_type, 1549
 - std::_result_type, 1550
 - std::_Maybe_unary_or_binary_function< _Res, _T1, _T2 >, 1550
 - std::_first_argument_type, 1551
 - std::_result_type, 1551
 - std::_second_argument_type, 1551
 - std::_Maybe_wrap_member_pointer, 1552
 - std::_Maybe_wrap_member_pointer< _Tp _Class::*, >, 1552
 - std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) const >, 1553
 - std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) const volatile >, 1554
 - std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) volatile >, 1555
 - std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) >, 1556
 - std::_Mu< _Arg, false, false >, 1557
 - std::_Mu< _Arg, false, true >, 1558
 - std::_Mu< _Arg, true, false >, 1558
 - std::_Mu< reference_wrapper< _Tp >, false, false >, 1559
 - std::_Placeholder, 1559
 - std::_Reference_wrapper_base, 1560
 - std::_Safe_tuple_element, 1561
 - std::_Safe_tuple_element_impl, 1561
 - std::_Safe_tuple_element_impl< __i, _Tuple, false >, 1562
 - std::_Temporary_buffer, 1563
 - std::_Temporary_buffer, 1564
 - std::_begin, 1564
 - std::_end, 1564
 - std::_requested_size, 1564
 - std::_size, 1565
 - std::_Tuple_impl< _Idx >, 1565
 - std::_Tuple_impl< _Idx, _Head, _Tail...>, 1566
 - std::_Vector_base, 1567
 - std::_Weak_result_type, 1568
 - std::_Weak_result_type_impl, 1569
 - std::_Weak_result_type_impl< _Res(*)(_ArgTypes...) >, 1570
 - std::_Weak_result_type_impl< _Res(&)(_ArgTypes...) >, 1569
 - std::_Weak_result_type_impl< _Res(_ArgTypes...) >, 1570
 - std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) const >, 1571
 - std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) const volatile >, 1571
 - std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) volatile >, 1572
 - std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) >, 1572
 - std::add_const, 1573
 - std::add_cv, 1573
 - std::add_lvalue_reference, 1573
 - std::add_pointer, 1574
 - std::add_rvalue_reference, 1574
 - std::add_volatile, 1575
 - std::adopt_lock_t, 1575
-

- std::aligned_storage, 1576
- std::alignment_of, 1577
- std::allocator, 1578
- std::allocator< void >, 1579
- std::allocator_arg_t, 1580
- std::array, 1580
- std::atomic, 1582
- std::atomic< _Tp * >, 1583
- std::atomic< bool >, 1585
- std::atomic< char >, 1586
- std::atomic< char16_t >, 1587
- std::atomic< char32_t >, 1588
- std::atomic< int >, 1588
- std::atomic< long >, 1589
- std::atomic< long long >, 1590
- std::atomic< short >, 1590
- std::atomic< signed char >, 1591
- std::atomic< unsigned char >, 1592
- std::atomic< unsigned int >, 1592
- std::atomic< unsigned long >, 1593
- std::atomic< unsigned long long >, 1594
- std::atomic< unsigned short >, 1594
- std::atomic< wchar_t >, 1595
- std::atomic_bool, 1596
- std::auto_ptr, 1597
 - ~auto_ptr, 1600
 - auto_ptr, 1599, 1600
 - element_type, 1599
 - get, 1601
 - operator*, 1601
 - operator->, 1601
 - operator=, 1601, 1602
 - release, 1602
 - reset, 1603
- std::auto_ptr_ref, 1603
- std::back_insert_iterator, 1604
 - back_insert_iterator, 1606
 - container_type, 1605
 - difference_type, 1605
 - iterator_category, 1605
 - operator*, 1607
 - operator++, 1607
 - operator=, 1607
 - pointer, 1606
 - reference, 1606
 - value_type, 1606
- std::bad_alloc, 1608
 - what, 1609
- std::bad_cast, 1609
 - what, 1610
- std::bad_exception, 1610
 - what, 1611
- std::bad_function_call, 1611
 - what, 1612
- std::bad_typeid, 1612
 - what, 1613
- std::bad_weak_ptr, 1613
 - what, 1614
- std::basic_filebuf, 1614
 - ~basic_filebuf, 1620
 - _M_buf, 1643
 - _M_buf_locale, 1643
 - _M_buf_size, 1643
 - _M_create_pback, 1620
 - _M_destroy_pback, 1620
 - _M_ext_buf, 1644
 - _M_ext_buf_size, 1644
 - _M_ext_next, 1644
 - _M_in_beg, 1644
 - _M_in_cur, 1645
 - _M_in_end, 1645
 - _M_mode, 1646
 - _M_out_beg, 1646
 - _M_out_cur, 1647
 - _M_out_end, 1647
 - _M_pback, 1648
 - _M_pback_cur_save, 1648
 - _M_pback_end_save, 1648
 - _M_pback_init, 1648
 - _M_reading, 1649
 - _M_set_buffer, 1621
 - __streambuf_type, 1618
- basic_filebuf, 1620
- char_type, 1618
- close, 1621
- eback, 1622
- egptr, 1622
- epptr, 1623
- gbump, 1623
- getloc, 1624
- gptr, 1624
- imbue, 1625

- in_avail, 1625
- int_type, 1618
- is_open, 1626
- off_type, 1619
- open, 1626, 1627
- overflow, 1627
- pbackfail, 1628
- pbase, 1629
- pbump, 1630
- pos_type, 1619
- pptr, 1630
- pubimbue, 1631
- pubseekoff, 1631
- pubseekpos, 1632
- pubsetbuf, 1632
- pubsync, 1632
- sbumpc, 1633
- seekoff, 1633
- seekpos, 1634
- setbuf, 1634
- setg, 1635
- setp, 1635
- sgetc, 1636
- sgetn, 1636
- showmanyc, 1637
- snextc, 1637
- sputbackc, 1638
- sputc, 1638
- sputn, 1639
- sungetc, 1639
- sync, 1639
- traits_type, 1619
- uflow, 1640
- underflow, 1640
- xsgetn, 1641
- xspn, 1642
- std::basic_fstream, 1649
 - ~basic_fstream, 1665
 - _M_gcount, 1715
 - _M_getloc, 1666
 - _M_write, 1666
 - __ctype_type, 1658
 - __num_get_type, 1658
 - __num_put_type, 1658
- adjustfield, 1715
- app, 1715
- ate, 1715
- bad, 1667
- badbit, 1716
- basefield, 1716
- basic_fstream, 1664, 1665
- beg, 1716
- binary, 1717
- boolalpha, 1717
- char_type, 1659
- clear, 1667
- close, 1667
- copyfmt, 1668
- cur, 1717
- dec, 1717
- end, 1718
- eof, 1668
- eofbit, 1718
- event, 1664
- event_callback, 1659
- exceptions, 1669
- fail, 1670
- failbit, 1718
- fill, 1670, 1671
- fixed, 1719
- flags, 1671, 1672
- floatfield, 1719
- flush, 1672
- fmtflags, 1660
- gcount, 1672
- get, 1673–1676
- getline, 1676, 1677
- getloc, 1678
- good, 1678
- goodbit, 1719
- hex, 1720
- ignore, 1678–1680
- imbue, 1680
- in, 1720
- init, 1681
- int_type, 1660, 1661
- internal, 1720
- iostate, 1661
- is_open, 1681
- iword, 1681
- left, 1721
- narrow, 1682

- oct, 1721
- off_type, 1661, 1662
- open, 1682, 1683
- openmode, 1662
- operator void *, 1683
- operator<<, 1684–1690
- operator>>, 1691–1698
- out, 1721
- peek, 1698
- pos_type, 1662, 1663
- precision, 1699
- put, 1699
- putback, 1700
- pword, 1701
- rdbuf, 1701, 1702
- rdstate, 1702
- read, 1703
- readsome, 1703
- register_callback, 1704
- right, 1721
- scientific, 1722
- seekdir, 1663
- seekg, 1704, 1705
- seekp, 1706
- setf, 1707
- setstate, 1708
- showbase, 1722
- showpoint, 1722
- showpos, 1722
- skipws, 1722
- sync, 1708
- sync_with_stdio, 1709
- tellg, 1709
- tellp, 1710
- tie, 1710, 1711
- traits_type, 1663, 1664
- trunc, 1723
- unget, 1711
- unitbuf, 1723
- unsetf, 1712
- uppercase, 1723
- widen, 1712
- width, 1713
- write, 1713
- xalloc, 1714
- std::basic_ifstream, 1723
- ~basic_ifstream, 1736
- _M_gcount, 1774
- _M_getloc, 1736
- __ctype_type, 1730
- __num_get_type, 1730
- __num_put_type, 1730
- adjustfield, 1775
- app, 1775
- ate, 1775
- bad, 1737
- badbit, 1775
- basefield, 1776
- basic_ifstream, 1735, 1736
- beg, 1776
- binary, 1776
- boolalpha, 1776
- char_type, 1731
- clear, 1737
- close, 1738
- copyfmt, 1738
- cur, 1777
- dec, 1777
- end, 1777
- eof, 1738
- eofbit, 1777
- event, 1735
- event_callback, 1731
- exceptions, 1739
- fail, 1740
- failbit, 1778
- fill, 1740, 1741
- fixed, 1778
- flags, 1741, 1742
- floatfield, 1779
- fmtflags, 1731
- gcount, 1742
- get, 1742–1745
- getline, 1746
- getloc, 1747
- good, 1748
- goodbit, 1779
- hex, 1779
- ignore, 1748, 1749
- imbue, 1750
- in, 1779
- init, 1750

int_type, 1732
internal, 1780
iostate, 1732
is_open, 1751
iword, 1751
left, 1780
narrow, 1751
oct, 1780
off_type, 1733
open, 1752
openmode, 1733
operator void *, 1753
operator >>, 1754–1760
out, 1780
peek, 1761
pos_type, 1734
precision, 1761
putback, 1762
pword, 1762
rdbuf, 1763
rdstate, 1764
read, 1764
readsome, 1765
register_callback, 1766
right, 1781
scientific, 1781
seekdir, 1734
seekg, 1766, 1767
setf, 1768
setstate, 1768
showbase, 1781
showpoint, 1781
showpos, 1782
skipws, 1782
sync, 1769
sync_with_stdio, 1770
tellg, 1770
tie, 1771
traits_type, 1734
trunc, 1782
unget, 1772
unitbuf, 1782
unsetf, 1772
uppercase, 1782
widen, 1772
width, 1773
xalloc, 1774
std::basic_ios, 1783
 ~basic_ios, 1793
 _M_getloc, 1793
 __ctype_type, 1787
 __num_get_type, 1787
 __num_put_type, 1787
adjustfield, 1810
app, 1810
ate, 1811
bad, 1794
badbit, 1811
basefield, 1811
basic_ios, 1793
beg, 1812
binary, 1812
boolalpha, 1812
char_type, 1788
clear, 1794
copyfmt, 1795
cur, 1812
dec, 1813
end, 1813
eof, 1795
eofbit, 1813
event, 1792
event_callback, 1788
exceptions, 1796
fail, 1797
failbit, 1813
fill, 1797
fixed, 1814
flags, 1798
floatfield, 1814
fmtflags, 1788
getloc, 1799
good, 1799
goodbit, 1814
hex, 1815
imbue, 1799
in, 1815
init, 1800
int_type, 1789
internal, 1815
iostate, 1790
iword, 1800

- left, 1816
- narrow, 1801
- oct, 1816
- off_type, 1790
- openmode, 1791
- operator void *, 1801
- out, 1816
- pos_type, 1791
- precision, 1802
- pword, 1803
- rdbuf, 1803, 1804
- rdstate, 1804
- register_callback, 1805
- right, 1816
- scientific, 1817
- seekdir, 1791
- setf, 1805, 1806
- setstate, 1806
- showbase, 1817
- showpoint, 1817
- showpos, 1817
- skipws, 1817
- sync_with_stdio, 1807
- tie, 1807, 1808
- traits_type, 1792
- trunc, 1818
- unitbuf, 1818
- unsetf, 1808
- uppercase, 1818
- widen, 1808
- width, 1809
- xalloc, 1810
- std::basic_istream, 1819
- ~basic_istream, 1833
- _M_gcount, 1881
- _M_getloc, 1834
- _M_write, 1834
- __ctype_type, 1826
- __num_get_type, 1826
- __num_put_type, 1827
- adjustfield, 1882
- app, 1882
- ate, 1882
- bad, 1835
- badbit, 1882
- basefield, 1883
- basic_istream, 1833
- beg, 1883
- binary, 1883
- boolalpha, 1883
- char_type, 1827
- clear, 1835
- copyfmt, 1835
- cur, 1884
- dec, 1884
- end, 1884
- eof, 1836
- eofbit, 1884
- event, 1833
- event_callback, 1828
- exceptions, 1836, 1837
- fail, 1838
- failbit, 1885
- fill, 1838
- fixed, 1885
- flags, 1839
- floatfield, 1886
- flush, 1840
- fmtflags, 1828
- gcount, 1840
- get, 1840–1843
- getline, 1844, 1845
- getloc, 1845
- good, 1846
- goodbit, 1886
- hex, 1886
- ignore, 1846, 1847
- imbue, 1848
- in, 1886
- init, 1848
- int_type, 1829
- internal, 1887
- iostate, 1830
- iword, 1849
- left, 1887
- narrow, 1849
- oct, 1887
- off_type, 1830
- openmode, 1831
- operator void *, 1850
- operator<<, 1850–1857
- operator>>, 1857–1864

- out, 1887
- peek, 1865
- pos_type, 1831
- precision, 1865
- put, 1866
- putback, 1866
- pword, 1867
- rdbuf, 1868
- rdstate, 1869
- read, 1869
- readsome, 1870
- register_callback, 1871
- right, 1888
- scientific, 1888
- seekdir, 1832
- seekg, 1871, 1872
- seekp, 1873
- setf, 1874
- setstate, 1874
- showbase, 1888
- showpoint, 1888
- showpos, 1889
- skipws, 1889
- sync, 1875
- sync_with_stdio, 1876
- tellg, 1876
- tellp, 1877
- tie, 1877
- traits_type, 1832
- trunc, 1889
- unget, 1878
- unitbuf, 1889
- unsetf, 1878
- uppercase, 1889
- widen, 1879
- width, 1879, 1880
- write, 1880
- xalloc, 1881
- std::basic_istream, 1890
 - ~basic_istream, 1901
 - _M_gcount, 1938
 - _M_getloc, 1902
 - __ctype_type, 1896
 - __num_get_type, 1896
 - __num_put_type, 1897
 - adjustfield, 1939
 - app, 1939
 - ate, 1939
 - bad, 1902
 - badbit, 1939
 - basefield, 1940
 - basic_istream, 1901
 - beg, 1940
 - binary, 1940
 - boolalpha, 1940
 - char_type, 1897
 - clear, 1902
 - copyfmt, 1903
 - cur, 1941
 - dec, 1941
 - end, 1941
 - eof, 1904
 - eofbit, 1941
 - event, 1901
 - event_callback, 1897
 - exceptions, 1904, 1905
 - fail, 1905
 - failbit, 1942
 - fill, 1906
 - fixed, 1942
 - flags, 1906, 1907
 - floatfield, 1943
 - fmtflags, 1897
 - gcount, 1907
 - get, 1907–1910
 - getline, 1911, 1912
 - getloc, 1912
 - good, 1913
 - goodbit, 1943
 - hex, 1943
 - ignore, 1913, 1914
 - imbue, 1915
 - in, 1943
 - init, 1915
 - int_type, 1898
 - internal, 1944
 - iostate, 1899
 - isword, 1916
 - left, 1944
 - narrow, 1916
 - oct, 1944
 - off_type, 1899

- openmode, 1899
- operator void *, 1917
- operator>>, 1917–1924
- out, 1944
- peek, 1924
- pos_type, 1900
- precision, 1925
- putback, 1925
- pword, 1926
- rdbuf, 1927
- rdstate, 1928
- read, 1928
- readsome, 1929
- register_callback, 1930
- right, 1945
- scientific, 1945
- seekdir, 1900
- seekg, 1930, 1931
- setf, 1932
- setstate, 1932
- showbase, 1945
- showpoint, 1945
- showpos, 1946
- skipws, 1946
- sync, 1933
- sync_with_stdio, 1934
- tellg, 1934
- tie, 1935
- traits_type, 1901
- trunc, 1946
- unget, 1936
- unitbuf, 1946
- unsetf, 1936
- uppercase, 1946
- widen, 1936
- width, 1937
- xalloc, 1938
- std::basic_istream::sentry, 1947
- operator bool, 1949
- sentry, 1948
- traits_type, 1948
- std::basic_istream, 1949
- ~basic_istream, 1962
- _M_gcount, 1999
- _M_getloc, 1962
- __ctype_type, 1956
- __num_get_type, 1956
- __num_put_type, 1956
- adjustfield, 2000
- app, 2000
- ate, 2000
- bad, 1962
- badbit, 2000
- basefield, 2001
- basic_istream, 1961
- beg, 2001
- binary, 2001
- boolalpha, 2001
- char_type, 1957
- clear, 1963
- copyfmt, 1963
- cur, 2002
- dec, 2002
- end, 2002
- eof, 1964
- eofbit, 2002
- event, 1961
- event_callback, 1957
- exceptions, 1964, 1965
- fail, 1965
- failbit, 2003
- fill, 1966
- fixed, 2003
- flags, 1967
- floatfield, 2004
- fmtflags, 1957
- gcount, 1967
- get, 1968–1971
- getline, 1971, 1972
- getloc, 1973
- good, 1973
- goodbit, 2004
- hex, 2004
- ignore, 1973–1975
- imbue, 1975
- in, 2004
- init, 1976
- int_type, 1958
- internal, 2005
- iostate, 1958
- isword, 1976
- left, 2005

- narrow, 1977
- oct, 2005
- off_type, 1959
- openmode, 1959
- operator void *, 1977
- operator>>, 1978–1984
- out, 2005
- peek, 1985
- pos_type, 1960
- precision, 1985, 1986
- putback, 1986
- pwd, 1987
- rdbuf, 1987, 1988
- rdstate, 1988
- read, 1989
- readsom, 1989
- register_callback, 1990
- right, 2006
- scientific, 2006
- seekdir, 1960
- seekg, 1990, 1991
- setf, 1992
- setstate, 1993
- showbase, 2006
- showpoint, 2006
- showpos, 2007
- skipws, 2007
- str, 1993, 1994
- sync, 1994
- sync_with_stdio, 1995
- tellg, 1995
- tie, 1996
- traits_type, 1960
- trunc, 2007
- unget, 1997
- unitbuf, 2007
- unsetf, 1997
- uppercase, 2007
- widen, 1997
- width, 1998
- xalloc, 1999
- std::basic_ofstream, 2008
- ~basic_ofstream, 2020
- _M_getloc, 2021
- _M_write, 2021
- __ctype_type, 2014
- __num_get_type, 2015
- __num_put_type, 2015
- adjustfield, 2050
- app, 2050
- ate, 2050
- bad, 2021
- badbit, 2050
- basefield, 2051
- basic_ofstream, 2019, 2020
- beg, 2051
- binary, 2051
- boolalpha, 2052
- char_type, 2015
- clear, 2022
- close, 2022
- copyfmt, 2023
- cur, 2052
- dec, 2052
- end, 2052
- eof, 2023
- eofbit, 2053
- event, 2019
- event_callback, 2015
- exceptions, 2024
- fail, 2025
- failbit, 2053
- fill, 2025, 2026
- fixed, 2053
- flags, 2026
- floatfield, 2054
- flush, 2027
- fmtflags, 2016
- getloc, 2027
- good, 2028
- goodbit, 2054
- hex, 2054
- imbue, 2028
- in, 2055
- init, 2029
- int_type, 2016
- internal, 2055
- iostate, 2017
- is_open, 2029
- iword, 2029
- left, 2055
- narrow, 2030

- oct, 2055
- off_type, 2017
- open, 2030, 2031
- openmode, 2017
- operator void *, 2031
- operator<<, 2032–2039
- out, 2056
- pos_type, 2018
- precision, 2039
- put, 2040
- pword, 2040
- rdbuf, 2041
- rdstate, 2042
- register_callback, 2042
- right, 2056
- scientific, 2056
- seekdir, 2018
- seekp, 2043
- setf, 2044
- setstate, 2045
- showbase, 2056
- showpoint, 2057
- showpos, 2057
- skipws, 2057
- sync_with_stdio, 2045
- tellp, 2046
- tie, 2046, 2047
- traits_type, 2018
- trunc, 2057
- unitbuf, 2057
- unsetf, 2047
- uppercase, 2058
- widen, 2047
- width, 2048
- write, 2049
- xalloc, 2049
- std::basic_ostream, 2058
 - ~basic_ostream, 2069
 - _M_getloc, 2069
 - _M_write, 2070
 - __ctype_type, 2064
 - __num_get_type, 2064
 - __num_put_type, 2064
 - adjustfield, 2097
 - app, 2097
 - ate, 2097
 - bad, 2070
 - badbit, 2098
 - basefield, 2098
 - basic_ostream, 2069
 - beg, 2098
 - binary, 2099
 - boolalpha, 2099
 - char_type, 2064
 - clear, 2071
 - copyfmt, 2071
 - cur, 2099
 - dec, 2099
 - end, 2100
 - eof, 2072
 - eofbit, 2100
 - event, 2069
 - event_callback, 2065
 - exceptions, 2072, 2073
 - fail, 2073
 - failbit, 2100
 - fill, 2074
 - fixed, 2101
 - flags, 2075
 - floatfield, 2101
 - flush, 2075
 - fmtflags, 2065
 - getloc, 2076
 - good, 2076
 - goodbit, 2101
 - hex, 2102
 - imbue, 2076
 - in, 2102
 - init, 2077
 - int_type, 2066
 - internal, 2102
 - iostate, 2066
 - isword, 2077
 - left, 2103
 - narrow, 2078
 - oct, 2103
 - off_type, 2067
 - openmode, 2067
 - operator void *, 2078
 - operator<<, 2079–2086
 - out, 2103
 - pos_type, 2067

- precision, 2086
- put, 2087
- pword, 2087
- rdbuf, 2088
- rdstate, 2089
- register_callback, 2090
- right, 2103
- scientific, 2104
- seekdir, 2068
- seekp, 2090, 2091
- setf, 2091
- setstate, 2092
- showbase, 2104
- showpoint, 2104
- showpos, 2104
- skipws, 2104
- sync_with_stdio, 2093
- tellp, 2093
- tie, 2093, 2094
- traits_type, 2068
- trunc, 2105
- unitbuf, 2105
- unsetf, 2094
- uppercase, 2105
- widen, 2095
- width, 2095, 2096
- write, 2096
- xalloc, 2097
- std::basic_ostream::sentry, 2105
 - ~sentry, 2106
 - operator bool, 2107
 - sentry, 2106
- std::basic_ostringstream, 2107
 - ~basic_ostringstream, 2119
 - _M_getloc, 2119
 - _M_write, 2120
 - __ctype_type, 2113
 - __num_get_type, 2114
 - __num_put_type, 2114
 - adjustfield, 2148
 - app, 2148
 - ate, 2148
 - bad, 2120
 - badbit, 2148
 - basefield, 2149
 - basic_ostringstream, 2118, 2119
 - beg, 2149
 - binary, 2149
 - boolalpha, 2149
 - char_type, 2114
 - clear, 2121
 - copyfmt, 2121
 - cur, 2150
 - dec, 2150
 - end, 2150
 - eof, 2122
 - eofbit, 2150
 - event, 2118
 - event_callback, 2114
 - exceptions, 2122, 2123
 - fail, 2123
 - failbit, 2151
 - fill, 2124
 - fixed, 2151
 - flags, 2125
 - floatfield, 2151
 - flush, 2125
 - fmtflags, 2115
 - getloc, 2126
 - good, 2126
 - goodbit, 2152
 - hex, 2152
 - imbue, 2126
 - in, 2152
 - init, 2127
 - int_type, 2116
 - internal, 2153
 - iostate, 2116
 - isword, 2127
 - left, 2153
 - narrow, 2128
 - oct, 2153
 - off_type, 2116
 - openmode, 2116
 - operator void *, 2128
 - operator<<, 2129–2136
 - out, 2153
 - pos_type, 2117
 - precision, 2136
 - put, 2137
 - pword, 2137
 - rdbuf, 2138

- rdstate, 2139
- register_callback, 2139
- right, 2154
- scientific, 2154
- seekdir, 2117
- seekp, 2140
- setf, 2141
- setstate, 2142
- showbase, 2154
- showpoint, 2154
- showpos, 2154
- skipws, 2155
- str, 2142, 2143
- sync_with_stdio, 2143
- tellp, 2143
- tie, 2144
- traits_type, 2117
- trunc, 2155
- unitbuf, 2155
- unsetf, 2145
- uppercase, 2155
- widen, 2145
- width, 2146
- write, 2146
- xalloc, 2147
- std::basic_regex, 2156
 - ~basic_regex, 2161
 - assign, 2161–2164
 - basic_regex, 2158–2160
 - flags, 2165
 - getloc, 2165
 - imbue, 2165
 - mark_count, 2165
 - operator=, 2166, 2167
 - swap, 2167
- std::basic_streambuf, 2168
 - ~basic_streambuf, 2174
 - _M_buf_locale, 2193
 - _M_in_beg, 2193
 - _M_in_cur, 2194
 - _M_in_end, 2194
 - _M_out_beg, 2195
 - _M_out_cur, 2195
 - _M_out_end, 2196
 - __streambuf_type, 2172
 - basic_streambuf, 2174
 - char_type, 2172
 - eback, 2174
 - egptr, 2175
 - epptr, 2175
 - gbump, 2176
 - getloc, 2176
 - gptr, 2177
 - imbue, 2177
 - in_avail, 2178
 - int_type, 2172
 - off_type, 2172
 - overflow, 2178
 - pbackfail, 2179
 - pbase, 2180
 - pbump, 2180
 - pos_type, 2173
 - pptr, 2181
 - pubimbue, 2181
 - pubseekoff, 2182
 - pubseekpos, 2182
 - pubsetbuf, 2183
 - pubsync, 2183
 - sbumpc, 2183
 - seekoff, 2184
 - seekpos, 2184
 - setbuf, 2185
 - setg, 2185
 - setp, 2186
 - sgetc, 2186
 - sgetn, 2187
 - showmanyc, 2187
 - snextc, 2188
 - sputbackc, 2188
 - sputc, 2189
 - sputn, 2189
 - sungetc, 2189
 - sync, 2190
 - traits_type, 2173
 - uflow, 2190
 - underflow, 2191
 - xsgetn, 2192
 - xspn, 2192
- std::basic_string, 2196
 - ~basic_string, 2206
 - append, 2206–2209
 - assign, 2209–2213

at, 2213, 2214
back, 2214
basic_string, 2202–2205
begin, 2215
c_str, 2215
capacity, 2215
cbegin, 2216
cend, 2216
clear, 2216
compare, 2216–2220
copy, 2220
crbegin, 2221
crend, 2221
data, 2221
empty, 2222
end, 2222
erase, 2222, 2223
find, 2224, 2225
find_first_not_of, 2226–2228
find_first_of, 2228–2230
find_last_not_of, 2230–2232
find_last_of, 2232–2234
front, 2234
get_allocator, 2235
insert, 2235–2240
length, 2240
max_size, 2240
npos, 2259
operator+=, 2241, 2242
operator=, 2242, 2243
push_back, 2245
rbegin, 2245
rend, 2245, 2246
replace, 2246–2252
reserve, 2253
resize, 2254
rfind, 2255, 2256
shrink_to_fit, 2257
size, 2257
substr, 2258
swap, 2258
std::basic_stringbuf, 2259
 _M_buf_locale, 2286
 _M_in_beg, 2286
 _M_in_cur, 2286
 _M_in_end, 2287
 _M_mode, 2287
 _M_out_beg, 2287
 _M_out_cur, 2288
 _M_out_end, 2288
 __streambuf_type, 2263
basic_stringbuf, 2265
char_type, 2263
eback, 2265
egptr, 2266
eptr, 2266
gbump, 2267
getloc, 2267
gptr, 2268
imbue, 2269
in_avail, 2269
int_type, 2263
off_type, 2264
overflow, 2269
pbackfail, 2270
pbase, 2271
pbump, 2272
pos_type, 2264
pptr, 2272
pubimbue, 2273
pubseekoff, 2273
pubseekpos, 2273
pubsetbuf, 2274
pubsync, 2274
sbumpc, 2275
seekoff, 2275
seekpos, 2276
setbuf, 2276
setg, 2277
setp, 2277
sgetc, 2278
sgetn, 2278
showmanyc, 2279
snextc, 2279
sputbackc, 2280
sputc, 2280
sputn, 2281
str, 2281
sungetc, 2282
sync, 2282
traits_type, 2264
uflow, 2283

- underflow, 2283
- xsgetn, 2284
- xspn, 2285
- std::basic_stringstream, 2289
 - ~basic_stringstream, 2305
 - _M_gcount, 2354
 - _M_getloc, 2306
 - _M_write, 2306
 - __ctype_type, 2298
 - __num_get_type, 2298
 - __num_put_type, 2298
- adjustfield, 2354
- app, 2354
- ate, 2354
- bad, 2306
- badbit, 2355
- basefield, 2355
- basic_stringstream, 2304, 2305
- beg, 2355
- binary, 2356
- boolalpha, 2356
- char_type, 2299
- clear, 2307
- copyfmt, 2307
- cur, 2356
- dec, 2356
- end, 2357
- eof, 2308
- eofbit, 2357
- event, 2304
- event_callback, 2299
- exceptions, 2308, 2309
- fail, 2309
- failbit, 2357
- fill, 2310
- fixed, 2358
- flags, 2311
- floatfield, 2358
- flush, 2311
- fmtflags, 2300
- gcount, 2312
- get, 2312–2315
- getline, 2316, 2317
- getloc, 2317
- good, 2318
- goodbit, 2358
- hex, 2359
- ignore, 2318, 2319
- imbue, 2320
- in, 2359
- init, 2320
- int_type, 2300, 2301
- internal, 2359
- iostate, 2301
- iword, 2321
- left, 2360
- narrow, 2321
- oct, 2360
- off_type, 2301, 2302
- openmode, 2302
- operator void *, 2322
- operator<<, 2322–2329
- operator>>, 2329–2336
- out, 2360
- peek, 2337
- pos_type, 2302, 2303
- precision, 2337
- put, 2338
- putback, 2338
- pword, 2339
- rdbuf, 2339, 2340
- rdstate, 2340
- read, 2341
- readsome, 2342
- register_callback, 2342
- right, 2360
- scientific, 2361
- seekdir, 2303
- seekg, 2343
- seekp, 2344, 2345
- setf, 2345, 2346
- setstate, 2346
- showbase, 2361
- showpoint, 2361
- showpos, 2361
- skipws, 2361
- str, 2347
- sync, 2347
- sync_with_stdio, 2348
- tellg, 2348
- tellp, 2349
- tie, 2349, 2350

- traits_type, 2303, 2304
- trunc, 2362
- unget, 2350
- unitbuf, 2362
- unsetf, 2351
- uppercase, 2362
- widen, 2351
- width, 2352
- write, 2352
- xalloc, 2353
- std::bernoulli_distribution, 2362
 - bernoulli_distribution, 2363
 - max, 2364
 - min, 2364
 - operator(), 2364
 - p, 2364
 - param, 2364, 2365
 - reset, 2365
 - result_type, 2363
- std::bernoulli_distribution::param_type, 2365
- std::bidirectional_iterator_tag, 2366
- std::binary_function, 2367
 - first_argument_type, 2368
 - result_type, 2368
 - second_argument_type, 2368
- std::binary_negate, 2369
 - first_argument_type, 2369
 - result_type, 2370
 - second_argument_type, 2370
- std::binder1st, 2370
 - argument_type, 2372
 - result_type, 2372
- std::binder2nd, 2372
 - argument_type, 2373
 - result_type, 2373
- std::binomial_distribution, 2374
 - max, 2375
 - min, 2375
 - operator<<, 2378
 - operator>>, 2378
 - operator(), 2376
 - operator==, 2378
 - p, 2376
 - param, 2376, 2377
 - reset, 2377
 - result_type, 2375
 - t, 2377
- std::binomial_distribution::param_type, 2379
- std::bitset, 2380
 - all, 2386
 - any, 2386
 - bitset, 2384, 2385
 - count, 2387
 - flip, 2387
 - none, 2387
 - operator<<, 2388
 - operator<=, 2388
 - operator>>, 2389
 - operator>=, 2389
 - operator~, 2391
 - operator^=, 2390
 - operator==, 2389
 - operator&=, 2388
 - reset, 2391
 - set, 2392
 - size, 2392
 - test, 2392
 - to_string, 2393
 - to_ulong, 2393
- std::bitset::reference, 2394
- std::cauchy_distribution, 2395
 - max, 2396
 - min, 2396
 - operator(), 2396
 - param, 2396, 2397
 - reset, 2397
 - result_type, 2396
- std::cauchy_distribution::param_type, 2397
- std::char_traits, 2398
- std::char_traits< __gnu_cxx::character< V, I, S > >, 2400
- std::char_traits< char >, 2401
- std::char_traits< wchar_t >, 2402
- std::chi_squared_distribution, 2403
 - max, 2405
 - min, 2405
 - operator<<, 2406
 - operator>>, 2407
 - operator(), 2405

- operator==, 2406
- param, 2405
- reset, 2406
- result_type, 2404
- std::chi_squared_distribution::param_type, 2407
- std::chrono, 782
 - duration_cast, 786
 - hours, 785
 - microseconds, 785
 - milliseconds, 785
 - minutes, 786
 - nanoseconds, 786
 - seconds, 786
 - time_point_cast, 786
- std::chrono::duration, 2408
- std::chrono::duration_values, 2409
- std::chrono::system_clock, 2410
- std::chrono::time_point, 2411
- std::chrono::treat_as_floating_point, 2412
- std::codecvt, 2413
 - do_out, 2415
 - in, 2415
 - out, 2416
 - unshift, 2417
- std::codecvt< _InternT, _ExternT, encoding_state >, 2418
 - do_out, 2421
 - in, 2421
 - out, 2422
 - unshift, 2423
- std::codecvt< char, char, mbstate_t >, 2424
 - do_out, 2426
 - in, 2426
 - out, 2427
 - unshift, 2428
- std::codecvt< wchar_t, char, mbstate_t >, 2429
 - do_out, 2431
 - in, 2431
 - out, 2432
 - unshift, 2433
- std::codecvt_base, 2434
- std::codecvt_byname, 2435
 - do_out, 2438
 - in, 2438
 - out, 2439
 - unshift, 2440
- std::collate, 2441
 - ~collate, 2445
 - char_type, 2444
 - collate, 2444
 - compare, 2445
 - do_compare, 2445
 - do_hash, 2446
 - do_transform, 2447
 - hash, 2447
 - id, 2448
 - string_type, 2444
 - transform, 2448
- std::collate_byname, 2449
 - char_type, 2451
 - compare, 2451
 - do_compare, 2451
 - do_hash, 2452
 - do_transform, 2453
 - hash, 2453
 - id, 2454
 - string_type, 2451
 - transform, 2454
- std::complex, 2455
 - complex, 2456
 - operator+=, 2456
 - operator=, 2456
 - value_type, 2456
- std::condition_variable, 2457
- std::condition_variable_any, 2458
- std::conditional, 2459
- std::const_mem_fun1_ref_t, 2459
 - first_argument_type, 2461
 - result_type, 2461
 - second_argument_type, 2461
- std::const_mem_fun1_t, 2461
 - first_argument_type, 2463
 - result_type, 2463
 - second_argument_type, 2463
- std::const_mem_fun_ref_t, 2463
 - argument_type, 2465
 - result_type, 2465
- std::const_mem_fun_t, 2465
 - argument_type, 2467

- result_type, 2467
- std::ctype, 2467
 - char_type, 2470
 - do_is, 2471
 - do_narrow, 2472
 - do_scan_is, 2473
 - do_scan_not, 2473
 - do_tolower, 2474
 - do_toupper, 2475
 - do_widen, 2476
 - id, 2483
 - is, 2477, 2478
 - narrow, 2478, 2479
 - scan_is, 2479
 - scan_not, 2480
 - tolower, 2480, 2481
 - toupper, 2481, 2482
 - widen, 2482, 2483
- std::ctype< char >, 2484
 - ~ctype, 2487
 - char_type, 2487
 - classic_table, 2488
 - ctype, 2487
 - do_narrow, 2488
 - do_tolower, 2489
 - do_toupper, 2490
 - do_widen, 2491
 - id, 2498
 - is, 2492
 - narrow, 2493
 - scan_is, 2494
 - scan_not, 2494
 - table, 2495
 - table_size, 2498
 - tolower, 2495, 2496
 - toupper, 2496, 2497
 - widen, 2497, 2498
- std::ctype< wchar_t >, 2499
 - ~ctype, 2503
 - char_type, 2503
 - ctype, 2503
 - do_is, 2504
 - do_narrow, 2504, 2505
 - do_scan_is, 2506
 - do_scan_not, 2506
 - do_tolower, 2507
 - do_toupper, 2508
 - do_widen, 2509
 - id, 2516
 - is, 2510
 - narrow, 2511
 - scan_is, 2512
 - scan_not, 2512
 - tolower, 2513
 - toupper, 2514
 - widen, 2515
- std::ctype_base, 2516
- std::ctype_byname, 2518
 - char_type, 2520
 - do_is, 2521
 - do_narrow, 2521, 2522
 - do_scan_is, 2523
 - do_scan_not, 2523
 - do_tolower, 2524
 - do_toupper, 2525
 - do_widen, 2526
 - id, 2533
 - is, 2527, 2528
 - narrow, 2528, 2529
 - scan_is, 2529
 - scan_not, 2530
 - tolower, 2530, 2531
 - toupper, 2531, 2532
 - widen, 2532, 2533
- std::ctype_byname< char >, 2534
 - char_type, 2536
 - classic_table, 2537
 - do_narrow, 2537
 - do_tolower, 2538
 - do_toupper, 2539
 - do_widen, 2540
 - id, 2547
 - is, 2541
 - narrow, 2542
 - scan_is, 2543
 - scan_not, 2543
 - table, 2544
 - table_size, 2547
 - tolower, 2544
 - toupper, 2545
 - widen, 2546
- std::decay, 2548

- std::decimal, 787
 - decimal32_to_long_long, 798
- std::decimal::decimal128, 2548
 - decimal128, 2550
- std::decimal::decimal32, 2550
 - decimal32, 2552
- std::decimal::decimal64, 2552
 - decimal64, 2554
- std::default_delete, 2554
- std::defer_lock_t, 2555
- std::deque, 2555
 - ~deque, 2565
 - _M_fill_initialize, 2565
 - _M_initialize_map, 2566
 - _M_new_elements_at_back, 2566
 - _M_new_elements_at_front, 2566
 - _M_pop_back_aux, 2567
 - _M_pop_front_aux, 2567
 - _M_push_back_aux, 2567
 - _M_push_front_aux, 2567
 - _M_range_check, 2568
 - _M_range_initialize, 2568
 - _M_reallocate_map, 2569
 - _M_reserve_elements_at_back, 2569
 - _M_reserve_elements_at_front, 2570
 - _M_reserve_map_at_back, 2570
 - _M_reserve_map_at_front, 2570
 - assign, 2570, 2571
 - at, 2572
 - back, 2573
 - begin, 2573
 - cbegin, 2574
 - cend, 2574
 - clear, 2574
 - crbegin, 2574
 - crend, 2575
 - deque, 2562–2564
 - emplace, 2575
 - empty, 2575
 - end, 2576
 - erase, 2576, 2577
 - front, 2577
 - get_allocator, 2578
 - insert, 2578–2580
 - max_size, 2580
 - operator=, 2580, 2581
 - pop_back, 2582
 - pop_front, 2583
 - push_back, 2583
 - push_front, 2584
 - rbegin, 2584
 - rend, 2584, 2585
 - resize, 2585
 - shrink_to_fit, 2586
 - size, 2586
 - swap, 2586
- std::discard_block_engine, 2587
 - base, 2590
 - discard, 2590
 - discard_block_engine, 2588–2590
 - max, 2590
 - min, 2591
 - operator<<, 2592
 - operator>>, 2593
 - operator(), 2591
 - operator==, 2592
 - result_type, 2588
 - seed, 2591, 2592
- std::discrete_distribution, 2594
 - max, 2595
 - min, 2595
 - operator<<, 2597
 - operator>>, 2597
 - operator(), 2595
 - param, 2596
 - probabilities, 2596
 - reset, 2597
 - result_type, 2595
- std::discrete_distribution::param_type, 2598
- std::divides, 2599
 - first_argument_type, 2600
 - result_type, 2600
 - second_argument_type, 2600
- std::domain_error, 2601
 - what, 2601
- std::enable_if, 2602
- std::enable_shared_from_this, 2602
- std::equal_to, 2603
 - first_argument_type, 2605

- result_type, 2605
- second_argument_type, 2605
- std::error_category, 2605
- std::error_code, 2606
- std::error_condition, 2607
- std::exception, 2607
 - what, 2609
- std::exponential_distribution, 2609
 - exponential_distribution, 2611
 - lambda, 2611
 - max, 2611
 - min, 2611
 - operator(), 2611
 - param, 2612
 - reset, 2612
 - result_type, 2610
- std::exponential_distribution::param_type, 2613
- std::extent, 2613
- std::extreme_value_distribution, 2615
 - a, 2616
 - b, 2616
 - max, 2616
 - min, 2617
 - operator(), 2617
 - param, 2617
 - reset, 2618
 - result_type, 2616
- std::extreme_value_distribution::param_type, 2618
- std::fisher_f_distribution, 2619
 - max, 2620
 - min, 2620
 - operator<=, 2622
 - operator>=, 2622
 - operator(), 2621
 - operator==, 2622
 - param, 2621
 - reset, 2621
 - result_type, 2620
- std::fisher_f_distribution::param_type, 2623
- std::forward_iterator_tag, 2624
- std::forward_list, 2625
 - ~forward_list, 2632
 - assign, 2632, 2633
 - before_begin, 2633, 2634
 - begin, 2634
 - cbegin, 2634
 - cbegin, 2635
 - cend, 2635
 - clear, 2635
 - emplace_after, 2635
 - emplace_front, 2636
 - empty, 2636
 - end, 2637
 - erase_after, 2637, 2638
 - forward_list, 2629–2631
 - front, 2638
 - get_allocator, 2639
 - insert_after, 2639–2641
 - max_size, 2641
 - merge, 2641, 2642
 - operator=, 2642, 2643
 - pop_front, 2643
 - push_front, 2644
 - remove, 2644
 - remove_if, 2644
 - resize, 2645
 - reverse, 2646
 - sort, 2646
 - splice_after, 2646, 2647
 - swap, 2648
 - unique, 2648, 2649
- std::fpos, 2649
 - fpos, 2650
 - operator streamoff, 2650
 - operator+, 2650
 - operator+=, 2651
 - operator-, 2651
 - operator=, 2651
 - state, 2651, 2652
- std::front_insert_iterator, 2652
 - container_type, 2653
 - difference_type, 2653
 - front_insert_iterator, 2655
 - iterator_category, 2654
 - operator*, 2655
 - operator++, 2655
 - operator=, 2655
 - pointer, 2654
 - reference, 2654

- value_type, 2654
- std::function< _Res(_ArgTypes...)>, 2656
 - function, 2658–2660
 - operator bool, 2660
 - operator(), 2660
 - operator=, 2661–2663
 - swap, 2663
 - target, 2663, 2664
 - target_type, 2664
- std::future, 2665
 - _M_get_result, 2667
 - future, 2666
 - get, 2667
- std::future< _Res & >, 2667
 - _M_get_result, 2670
 - future, 2669
 - get, 2670
- std::future< void >, 2670
 - _M_get_result, 2673
 - future, 2672
 - get, 2673
- std::future_error, 2673
 - what, 2674
- std::gamma_distribution, 2675
 - alpha, 2677
 - beta, 2677
 - gamma_distribution, 2677
 - max, 2677
 - min, 2677
 - operator<=, 2679
 - operator>>, 2680
 - operator(), 2678
 - operator==, 2679
 - param, 2678
 - reset, 2679
 - result_type, 2676
- std::gamma_distribution::param_type, 2680
- std::geometric_distribution, 2681
 - max, 2682
 - min, 2683
 - operator(), 2683
 - p, 2683
 - param, 2683, 2684
 - reset, 2684
 - result_type, 2682
- std::geometric_distribution::param_type, 2684
- std::greater, 2685
 - first_argument_type, 2686
 - result_type, 2686
 - second_argument_type, 2686
- std::greater_equal, 2687
 - first_argument_type, 2688
 - result_type, 2688
 - second_argument_type, 2688
- std::gslice, 2688
- std::gslice_array, 2689
- std::has_nothrow_copy_assign, 2691
- std::has_nothrow_copy_constructor, 2692
- std::has_nothrow_default_constructor, 2694
- std::has_trivial_copy_assign, 2695
- std::has_trivial_copy_constructor, 2696
- std::has_trivial_default_constructor, 2698
- std::has_trivial_destructor, 2699
- std::has_virtual_destructor, 2700
- std::hash, 2702
- std::hash< __debug::bitset< _Nb > >, 2703
- std::hash< __debug::vector< bool, _Alloc > >, 2704
- std::hash< __gnu_cxx::throw_value_limit >, 2704
 - argument_type, 2706
 - result_type, 2706
- std::hash< __gnu_cxx::throw_value_random >, 2706
 - argument_type, 2708
 - result_type, 2708
- std::hash< __profile::bitset< _Nb > >, 2708
- std::hash< __profile::vector< bool, _Alloc > >, 2709
- std::hash< __shared_ptr< _Tp, _Lp > >, 2709
 - argument_type, 2711
 - result_type, 2711
- std::hash< _Tp * >, 2711
- std::hash< error_code >, 2712
- std::hash< shared_ptr< _Tp > >, 2712

- argument_type, 2714
- result_type, 2714
- std::hash< string >, 2714
- std::hash< thread::id >, 2715
- std::hash< type_index >, 2715
- std::hash< u16string >, 2716
- std::hash< u32string >, 2716
- std::hash< unique_ptr< _Tp, _Dp > >, 2717
 - argument_type, 2719
 - result_type, 2719
- std::hash< wstring >, 2719
- std::hash<::bitset< _Nb > >, 2720
- std::hash<::vector< bool, _Alloc > >, 2720
- std::independent_bits_engine, 2721
 - base, 2724
 - discard, 2724
 - independent_bits_engine, 2722–2724
 - max, 2725
 - min, 2725
 - operator>>, 2727
 - operator(), 2725
 - operator==, 2726
 - result_type, 2722
 - seed, 2725, 2726
- std::indirect_array, 2727
 - operator<=, 2730
 - operator>=, 2730
 - operator*, 2729
 - operator^=, 2730
 - operator+=, 2729
 - operator-=, 2730
 - operator/=, 2730
 - operator%=, 2729
 - operator&=, 2729
- std::initializer_list, 2731
- std::input_iterator_tag, 2732
- std::insert_iterator, 2732
 - container_type, 2734
 - difference_type, 2734
 - insert_iterator, 2735
 - iterator_category, 2734
 - operator*, 2735
 - operator++, 2735, 2736
 - operator=, 2736
 - pointer, 2734
 - reference, 2735
 - value_type, 2735
- std::integral_constant, 2737
- std::invalid_argument, 2738
 - what, 2739
- std::ios_base, 2739
 - ~ios_base, 2745
 - _M_getloc, 2745
 - adjustfield, 2752
 - app, 2752
 - ate, 2752
 - badbit, 2752
 - basefield, 2753
 - beg, 2753
 - binary, 2753
 - boolalpha, 2754
 - cur, 2754
 - dec, 2754
 - end, 2754
 - eofbit, 2755
 - event, 2745
 - event_callback, 2742
 - failbit, 2755
 - fixed, 2755
 - flags, 2746
 - floatfield, 2756
 - fmtflags, 2742
 - getloc, 2746
 - goodbit, 2756
 - hex, 2756
 - imbue, 2747
 - in, 2757
 - internal, 2757
 - iostate, 2743
 - word, 2747
 - left, 2757
 - oct, 2757
 - openmode, 2744
 - out, 2758
 - precision, 2747, 2748
 - pwd, 2748
 - register_callback, 2749
 - right, 2758
 - scientific, 2758

- seekdir, 2744
- setf, 2749
- showbase, 2758
- showpoint, 2759
- showpos, 2759
- skipws, 2759
- sync_with_stdio, 2750
- trunc, 2759
- unitbuf, 2759
- unsetf, 2750
- uppercase, 2760
- width, 2751
- xalloc, 2751
- std::ios_base::failure, 2760
 - what, 2761
- std::is_abstract, 2761
- std::is_arithmetic, 2763
- std::is_array, 2764
- std::is_base_of, 2765
- std::is_bind_expression, 2766
- std::is_bind_expression< _Bind< _-
Signature > >, 2767
- std::is_bind_expression< _Bind_result<
_Result, _Signature > >, 2768
- std::is_class, 2769
- std::is_compound, 2771
- std::is_const, 2772
- std::is_constructible, 2773
- std::is_convertible, 2774
- std::is_empty, 2775
- std::is_enum, 2776
- std::is_error_code_enum, 2778
- std::is_error_code_enum< future_errc >,
2779
- std::is_error_condition_enum, 2780
- std::is_explicitly_convertible, 2781
- std::is_floating_point, 2782
- std::is_function, 2783
- std::is_fundamental, 2784
- std::is_integral, 2785
- std::is_literal_type, 2785
- std::is_lvalue_reference, 2787
- std::is_member_function_pointer, 2788
- std::is_member_object_pointer, 2789
- std::is_nothrow_constructible, 2789
- std::is_object, 2790
- std::is_placeholder, 2791
- std::is_placeholder< _Placeholder< _-
Num > >, 2793
- std::is_pod, 2794
- std::is_pointer, 2795
- std::is_polymorphic, 2796
- std::is_reference, 2797
- std::is_rvalue_reference, 2798
- std::is_same, 2799
- std::is_scalar, 2800
- std::is_signed, 2801
- std::is_standard_layout, 2802
- std::is_trivial, 2803
- std::is_union, 2805
- std::is_unsigned, 2806
- std::is_void, 2807
- std::is_volatile, 2807
- std::istream_iterator, 2809
 - difference_type, 2810
 - istream_iterator, 2811
 - iterator_category, 2810
 - pointer, 2810
 - reference, 2810
 - value_type, 2810
- std::istreambuf_iterator, 2811
 - char_type, 2813
 - difference_type, 2813
 - equal, 2816
 - int_type, 2813
 - istream_type, 2813
 - istreambuf_iterator, 2815
 - iterator_category, 2814
 - operator*, 2816
 - operator++, 2816
 - pointer, 2814
 - reference, 2814
 - streambuf_type, 2814
 - traits_type, 2814
 - value_type, 2815
- std::iterator, 2817
 - difference_type, 2818
 - iterator_category, 2818
 - pointer, 2818
 - reference, 2818
 - value_type, 2819
- std::iterator_traits< _Tp * >, 2819

- `std::iterator_traits< const _Tp * >`, 2819
- `std::length_error`, 2820
 - `what`, 2821
- `std::less`, 2821
 - `first_argument_type`, 2823
 - `result_type`, 2823
 - `second_argument_type`, 2823
- `std::less_equal`, 2823
 - `first_argument_type`, 2825
 - `result_type`, 2825
 - `second_argument_type`, 2825
- `std::linear_congruential_engine`, 2825
 - `discard`, 2828
 - `increment`, 2832
 - `linear_congruential_engine`, 2827, 2828
 - `max`, 2828
 - `min`, 2828
 - `modulus`, 2832
 - `multiplier`, 2832
 - `operator<<`, 2830
 - `operator>>`, 2831
 - `operator()`, 2829
 - `operator==`, 2830
 - `result_type`, 2827
 - `seed`, 2829
- `std::list`, 2832
 - `_M_create_node`, 2840
 - `assign`, 2840, 2841
 - `back`, 2842
 - `begin`, 2842
 - `cbegin`, 2842
 - `cend`, 2843
 - `clear`, 2843
 - `crbegin`, 2843
 - `crend`, 2843
 - `emplace`, 2844
 - `empty`, 2844
 - `end`, 2844, 2845
 - `erase`, 2845
 - `front`, 2846
 - `get_allocator`, 2846
 - `insert`, 2847–2849
 - `list`, 2837–2839
 - `max_size`, 2849
 - `merge`, 2849, 2850
 - `operator=`, 2850, 2851
 - `pop_back`, 2852
 - `pop_front`, 2852
 - `push_back`, 2852
 - `push_front`, 2853
 - `rbegin`, 2853
 - `remove`, 2854
 - `remove_if`, 2854
 - `rend`, 2854, 2855
 - `resize`, 2855
 - `reverse`, 2856
 - `size`, 2856
 - `sort`, 2856
 - `splice`, 2857, 2858
 - `swap`, 2858
 - `unique`, 2859
- `std::locale`, 2860
 - `~locale`, 2864
 - `all`, 2868
 - `category`, 2862
 - `classic`, 2864
 - `collate`, 2868
 - `combine`, 2864
 - `ctype`, 2868
 - `global`, 2864
 - `has_facet`, 2867
 - `locale`, 2862, 2863
 - `messages`, 2868
 - `monetary`, 2869
 - `name`, 2865
 - `none`, 2869
 - `numeric`, 2869
 - `operator()`, 2865
 - `operator=`, 2866
 - `operator==`, 2866
 - `time`, 2870
 - `use_facet`, 2867
- `std::locale::facet`, 2870
 - `~facet`, 2872
 - `facet`, 2872
- `std::locale::id`, 2872
 - `has_facet`, 2873
 - `id`, 2873
 - `use_facet`, 2874
- `std::lock_guard`, 2874
- `std::logic_error`, 2875

- logic_error, 2876
- what, 2876
- std::logical_and, 2876
 - first_argument_type, 2878
 - result_type, 2878
 - second_argument_type, 2878
- std::logical_not, 2878
 - argument_type, 2880
 - result_type, 2880
- std::logical_or, 2880
 - first_argument_type, 2882
 - result_type, 2882
 - second_argument_type, 2882
- std::lognormal_distribution, 2882
 - max, 2884
 - min, 2884
 - operator<=, 2885
 - operator>=, 2886
 - operator(), 2884
 - operator==, 2886
 - param, 2884, 2885
 - reset, 2885
 - result_type, 2884
- std::lognormal_distribution::param_type, 2887
- std::make_signed, 2887
- std::make_unsigned, 2888
- std::map, 2888
 - at, 2894
 - begin, 2894
 - cbegin, 2895
 - cend, 2895
 - clear, 2895
 - count, 2895
 - crbegin, 2896
 - crend, 2896
 - empty, 2896
 - end, 2897
 - equal_range, 2897, 2898
 - erase, 2898, 2899
 - find, 2900
 - get_allocator, 2901
 - insert, 2901–2903
 - key_comp, 2903
 - lower_bound, 2904
 - map, 2891–2893
 - max_size, 2905
 - operator=, 2905, 2906
 - rbegin, 2907
 - rend, 2907
 - size, 2908
 - swap, 2908
 - upper_bound, 2908, 2909
 - value_comp, 2909
- std::mask_array, 2910
 - operator<=, 2912
 - operator>=, 2912
 - operator*=, 2912
 - operator^=, 2913
 - operator+=, 2912
 - operator=, 2912
 - operator/=, 2912
 - operator%=, 2911
 - operator&=, 2911
- std::match_results, 2913
 - ~match_results, 2918
 - begin, 2918
 - cbegin, 2919
 - cend, 2919
 - empty, 2919
 - end, 2920
 - format, 2920
 - get_allocator, 2920
 - length, 2921
 - match_results, 2918
 - max_size, 2921
 - operator=, 2921
 - position, 2922
 - prefix, 2923
 - size, 2923
 - str, 2923
 - suffix, 2924
 - swap, 2924
- std::mem_fun1_ref_t, 2925
 - first_argument_type, 2926
 - result_type, 2926
 - second_argument_type, 2926
- std::mem_fun1_t, 2926
 - first_argument_type, 2928
 - result_type, 2928
 - second_argument_type, 2928
- std::mem_fun_ref_t, 2928

- argument_type, 2930
- result_type, 2930
- std::mem_fun_t, 2930
 - argument_type, 2932
 - result_type, 2932
- std::messages, 2932
 - ~messages, 2936
 - char_type, 2935
 - id, 2936
 - messages, 2935
 - string_type, 2935
- std::messages_base, 2936
- std::messages_byname, 2937
 - char_type, 2940
 - id, 2940
 - string_type, 2940
- std::minus, 2940
 - first_argument_type, 2942
 - result_type, 2942
 - second_argument_type, 2942
- std::modulus, 2942
 - first_argument_type, 2944
 - result_type, 2944
 - second_argument_type, 2944
- std::money_base, 2944
- std::money_get, 2946
 - ~money_get, 2949
 - char_type, 2948
 - do_get, 2949
 - get, 2950
 - id, 2951
 - iter_type, 2948
 - money_get, 2948
 - string_type, 2948
- std::money_put, 2952
 - ~money_put, 2954
 - char_type, 2954
 - do_put, 2955
 - id, 2958
 - iter_type, 2954
 - money_put, 2954
 - put, 2956, 2957
 - string_type, 2954
- std::moneypunct, 2958
 - ~moneypunct, 2962
 - char_type, 2961
 - curr_symbol, 2963
 - decimal_point, 2963
 - do_curr_symbol, 2963
 - do_decimal_point, 2964
 - do_frac_digits, 2964
 - do_grouping, 2965
 - do_neg_format, 2965
 - do_negative_sign, 2965
 - do_pos_format, 2966
 - do_positive_sign, 2966
 - do_thousands_sep, 2967
 - frac_digits, 2967
 - grouping, 2968
 - id, 2971
 - intl, 2971
 - moneypunct, 2961, 2962
 - neg_format, 2968
 - negative_sign, 2969
 - pos_format, 2969
 - positive_sign, 2970
 - string_type, 2961
 - thousands_sep, 2971
- std::moneypunct_byname, 2972
 - char_type, 2974
 - curr_symbol, 2975
 - decimal_point, 2975
 - do_curr_symbol, 2975
 - do_decimal_point, 2976
 - do_frac_digits, 2976
 - do_grouping, 2977
 - do_neg_format, 2977
 - do_negative_sign, 2977
 - do_pos_format, 2978
 - do_positive_sign, 2978
 - do_thousands_sep, 2979
 - frac_digits, 2979
 - grouping, 2980
 - id, 2983
 - intl, 2983
 - neg_format, 2980
 - negative_sign, 2981
 - pos_format, 2981
 - positive_sign, 2982
 - string_type, 2974
 - thousands_sep, 2983
- std::move_iterator, 2984

- std::multimap, 2985
 - begin, 2990
 - cbegin, 2991
 - cend, 2991
 - clear, 2991
 - count, 2991
 - crbegin, 2992
 - crend, 2992
 - empty, 2992
 - end, 2992, 2993
 - equal_range, 2993
 - erase, 2994, 2995
 - find, 2996
 - get_allocator, 2997
 - insert, 2997, 2998
 - key_comp, 2999
 - lower_bound, 2999, 3000
 - max_size, 3000
 - multimap, 2988–2990
 - operator=, 3000, 3001
 - rbegin, 3002
 - rend, 3002
 - size, 3003
 - swap, 3003
 - upper_bound, 3003, 3004
 - value_comp, 3004
- std::multiplies, 3004
 - first_argument_type, 3006
 - result_type, 3006
 - second_argument_type, 3006
- std::multiset, 3006
 - begin, 3011
 - cbegin, 3012
 - cend, 3012
 - clear, 3012
 - count, 3012
 - crbegin, 3013
 - crend, 3013
 - empty, 3013
 - end, 3013
 - equal_range, 3013, 3014
 - erase, 3015, 3016
 - find, 3016, 3017
 - get_allocator, 3017
 - insert, 3017–3019
 - key_comp, 3019
 - lower_bound, 3019, 3020
 - max_size, 3020
 - multiset, 3009–3011
 - operator<, 3024
 - operator=, 3021
 - operator==, 3025
 - rbegin, 3022
 - rend, 3022
 - size, 3022
 - swap, 3023
 - upper_bound, 3023
 - value_comp, 3024
- std::mutex, 3025
- std::negate, 3026
 - argument_type, 3027
 - result_type, 3027
- std::negative_binomial_distribution, 3027
 - k, 3029
 - max, 3029
 - min, 3029
 - operator<<, 3031
 - operator>>, 3032
 - operator(), 3030
 - operator==, 3031
 - p, 3030
 - param, 3030
 - reset, 3031
 - result_type, 3029
- std::negative_binomial_distribution::param_type, 3032
- std::nested_exception, 3033
- std::normal_distribution, 3034
 - max, 3036
 - mean, 3036
 - min, 3036
 - normal_distribution, 3036
 - operator<<, 3038
 - operator>>, 3039
 - operator(), 3036, 3037
 - operator==, 3039
 - param, 3037
 - reset, 3038
 - result_type, 3035
 - stddev, 3038

- std::normal_distribution::param_type, 3039
- std::not_equal_to, 3040
 - first_argument_type, 3042
 - result_type, 3042
 - second_argument_type, 3042
- std::num_get, 3042
 - ~num_get, 3046
 - char_type, 3045
 - do_get, 3046–3053
 - get, 3053–3062
 - id, 3063
 - iter_type, 3045
 - num_get, 3046
- std::num_put, 3063
 - ~num_put, 3066
 - char_type, 3066
 - do_put, 3067–3070
 - id, 3078
 - iter_type, 3066
 - num_put, 3066
 - put, 3071–3077
- std::numeric_limits, 3078
 - denorm_min, 3080
 - digits, 3082
 - digits10, 3082
 - epsilon, 3080
 - has_denorm, 3082
 - has_denorm_loss, 3082
 - has_infinity, 3083
 - has_quiet_NaN, 3083
 - has_signaling_NaN, 3083
 - infinity, 3080
 - is_bounded, 3083
 - is_exact, 3083
 - is_iec559, 3083
 - is_integer, 3084
 - is_modulo, 3084
 - is_signed, 3084
 - is_specialized, 3084
 - lowest, 3081
 - max, 3081
 - max_digits10, 3084
 - max_exponent, 3085
 - max_exponent10, 3085
 - min, 3081
 - min_exponent, 3085
 - min_exponent10, 3085
 - quiet_NaN, 3081
 - radix, 3085
 - round_error, 3081
 - round_style, 3085
 - signaling_NaN, 3082
 - tinyness_before, 3086
 - traps, 3086
- std::numeric_limits< bool >, 3086
- std::numeric_limits< char >, 3087
- std::numeric_limits< char16_t >, 3089
- std::numeric_limits< char32_t >, 3090
- std::numeric_limits< double >, 3091
- std::numeric_limits< float >, 3093
- std::numeric_limits< int >, 3094
- std::numeric_limits< long >, 3095
- std::numeric_limits< long double >, 3097
- std::numeric_limits< long long >, 3098
- std::numeric_limits< short >, 3099
- std::numeric_limits< signed char >, 3101
- std::numeric_limits< unsigned char >, 3102
- std::numeric_limits< unsigned int >, 3103
- std::numeric_limits< unsigned long >, 3105
- std::numeric_limits< unsigned long long >, 3106
- std::numeric_limits< unsigned short >, 3107
- std::numeric_limits< wchar_t >, 3109
- std::numpunct, 3110
 - ~numpunct, 3114
 - char_type, 3113
 - decimal_point, 3114
 - do_decimal_point, 3115
 - do_falsename, 3115
 - do_grouping, 3115
 - do_thousands_sep, 3116
 - do_truename, 3116
 - falsename, 3117
 - grouping, 3117
 - id, 3119
 - numpunct, 3113, 3114

- string_type, 3113
- thousands_sep, 3118
- truename, 3118
- std::numpunct_byname, 3119
 - char_type, 3121
 - decimal_point, 3121
 - do_decimal_point, 3122
 - do_falsename, 3122
 - do_grouping, 3122
 - do_thousands_sep, 3123
 - do_truename, 3123
 - falsename, 3124
 - grouping, 3124
 - id, 3126
 - string_type, 3121
 - thousands_sep, 3125
 - truename, 3125
- std::once_flag, 3126
 - call_once, 3126
- std::ostream_iterator, 3127
 - char_type, 3128
 - difference_type, 3128
 - iterator_category, 3129
 - operator=, 3131
 - ostream_iterator, 3130
 - ostream_type, 3129
 - pointer, 3129
 - reference, 3129
 - traits_type, 3129
 - value_type, 3130
- std::ostreambuf_iterator, 3131
 - char_type, 3133
 - difference_type, 3133
 - failed, 3135
 - iterator_category, 3133
 - operator*, 3135
 - operator++, 3136
 - operator=, 3136
 - ostream_type, 3133
 - ostreambuf_iterator, 3135
 - pointer, 3134
 - reference, 3134
 - streambuf_type, 3134
 - traits_type, 3134
 - value_type, 3134
- std::out_of_range, 3137
 - what, 3137
- std::output_iterator_tag, 3138
- std::overflow_error, 3138
 - what, 3139
- std::owner_less< shared_ptr< _Tp > >, 3139
 - first_argument_type, 3140
 - result_type, 3140
 - second_argument_type, 3140
- std::owner_less< weak_ptr< _Tp > >, 3140
 - first_argument_type, 3141
 - result_type, 3141
 - second_argument_type, 3141
- std::packaged_task< _Res(_ArgTypes...)>, 3142
- std::pair, 3143
 - first, 3145
 - pair, 3144, 3145
 - second, 3145
 - second_type, 3144
- std::piecewise_constant_distribution, 3146
 - densities, 3147
 - intervals, 3147
 - max, 3148
 - min, 3148
 - operator<<, 3149
 - operator>>, 3150
 - operator(), 3148
 - param, 3148, 3149
 - reset, 3149
 - result_type, 3147
- std::piecewise_constant_distribution::param_type, 3150
- std::piecewise_construct_t, 3151
- std::piecewise_linear_distribution, 3152
 - densities, 3153
 - intervals, 3153
 - max, 3154
 - min, 3154
 - operator<<, 3156
 - operator>>, 3156
 - operator(), 3154
 - param, 3155

- reset, 3155
- result_type, 3153
- std::piecewise_linear_
 - distribution::param_type, 3157
- std::placeholders, 798
- std::plus, 3158
 - first_argument_type, 3159
 - result_type, 3159
 - second_argument_type, 3159
- std::pointer_to_binary_function, 3159
 - first_argument_type, 3161
 - result_type, 3161
 - second_argument_type, 3161
- std::pointer_to_unary_function, 3161
 - argument_type, 3163
 - result_type, 3163
- std::poisson_distribution, 3163
 - max, 3165
 - mean, 3165
 - min, 3165
 - operator<<, 3167
 - operator>>, 3167
 - operator(), 3165
 - operator==, 3167
 - param, 3166
 - reset, 3166
 - result_type, 3164
- std::poisson_distribution::param_type, 3168
- std::priority_queue, 3169
 - empty, 3171
 - pop, 3171
 - priority_queue, 3170, 3171
 - push, 3172
 - size, 3172
 - top, 3173
- std::promise, 3173
- std::promise<_Res & >, 3174
- std::promise< void >, 3175
- std::queue, 3175
 - back, 3177
 - c, 3179
 - empty, 3178
 - front, 3178
 - pop, 3178
 - push, 3179
 - queue, 3177
 - size, 3179
- std::random_access_iterator_tag, 3180
- std::random_device, 3181
 - result_type, 3181
- std::range_error, 3182
 - what, 3182
- std::rank, 3183
- std::ratio, 3184
- std::ratio_add, 3185
- std::ratio_divide, 3185
- std::ratio_equal, 3186
- std::ratio_multiply, 3187
- std::ratio_not_equal, 3188
- std::ratio_subtract, 3189
- std::raw_storage_iterator, 3190
 - difference_type, 3191
 - iterator_category, 3191
 - pointer, 3191
 - reference, 3191
 - value_type, 3192
- std::recursive_mutex, 3192
- std::recursive_timed_mutex, 3193
- std::reference_wrapper, 3193
- std::regex_constants, 799
 - __match_flag, 801
 - __syntax_option, 802
- awk, 804
- basic, 804
- collate, 804
- ECMAScript, 804
- egrep, 805
- error_backref, 802
- error_badbrace, 802
- error_badrepeat, 802
- error_brace, 802
- error_brack, 803
- error_collate, 803
- error_complexity, 803
- error_ctype, 803
- error_escape, 803
- error_paren, 803
- error_range, 803
- error_space, 803
- error_stack, 804

- error_type, 802
- extended, 805
- format_default, 805
- format_first_only, 806
- format_no_copy, 806
- format_sed, 806
- grep, 806
- icase, 806
- match_any, 807
- match_continuous, 807
- match_default, 807
- match_flag_type, 801
- match_not_bol, 807
- match_not_bow, 807
- match_not_eol, 807
- match_not_eow, 808
- match_not_null, 808
- match_prev_avail, 808
- nosubs, 808
- optimize, 808
- syntax_option_type, 801
- std::regex_error, 3195
 - code, 3196
 - regex_error, 3196
 - what, 3196
- std::regex_iterator, 3197
 - operator*, 3199
 - operator++, 3199, 3200
 - operator->, 3200
 - operator=, 3200
 - operator==, 3200
 - regex_iterator, 3198
- std::regex_token_iterator, 3201
 - operator*, 3205
 - operator++, 3205, 3206
 - operator->, 3206
 - operator=, 3206
 - operator==, 3207
 - regex_token_iterator, 3202–3204
- std::regex_traits, 3207
 - getloc, 3209
 - imbue, 3209
 - length, 3209
 - lookup_classname, 3210
 - lookup_collatename, 3211
 - regex_traits, 3208
 - transform, 3211
 - transform_primary, 3212
 - translate, 3213
 - translate_nocase, 3213
- std::rel_ops, 809
 - operator<=, 809
 - operator>, 810
 - operator>=, 810
- std::remove_all_extents, 3214
- std::remove_const, 3214
- std::remove_cv, 3215
- std::remove_extent, 3215
- std::remove_pointer, 3216
- std::remove_reference, 3216
- std::remove_volatile, 3216
- std::reverse_iterator, 3217
 - base, 3221
 - difference_type, 3219
 - iterator_category, 3219
 - operator*, 3221
 - operator+, 3221
 - operator++, 3222
 - operator+=, 3222
 - operator-, 3223
 - operator->, 3224
 - operator--, 3223
 - operator=, 3224
 - pointer, 3219
 - reference, 3219
 - reverse_iterator, 3220
 - value_type, 3220
- std::runtime_error, 3225
 - runtime_error, 3226
 - what, 3226
- std::seed_seq, 3226
 - result_type, 3227
 - seed_seq, 3227
- std::set, 3228
 - allocator_type, 3231
 - begin, 3237
 - cbegin, 3237
 - cend, 3237
 - clear, 3238
 - const_iterator, 3231
 - const_pointer, 3231
 - const_reference, 3231

- const_reverse_iterator, 3231
- count, 3238
- crbegin, 3238
- crend, 3239
- difference_type, 3232
- empty, 3239
- end, 3239
- equal_range, 3239, 3240
- erase, 3240, 3241
- find, 3242
- get_allocator, 3243
- insert, 3243–3245
- iterator, 3232
- key_comp, 3245
- key_compare, 3232
- key_type, 3232
- lower_bound, 3245, 3246
- max_size, 3246
- operator=, 3246, 3247
- pointer, 3233
- rbegin, 3248
- reference, 3233
- rend, 3248
- reverse_iterator, 3233
- set, 3234–3236
- size, 3248
- size_type, 3233
- swap, 3248
- upper_bound, 3249
- value_comp, 3249
- value_compare, 3234
- value_type, 3234
- std::shared_future, 3250
 - _M_get_result, 3252
 - get, 3252
 - shared_future, 3251, 3252
- std::shared_future< _Res & >, 3253
 - _M_get_result, 3255
 - get, 3255
 - shared_future, 3254
- std::shared_future< void >, 3255
 - _M_get_result, 3258
 - shared_future, 3257
- std::shared_ptr, 3258
 - allocate_shared, 3266
 - shared_ptr, 3261–3266
- std::shuffle_order_engine, 3267
 - base, 3270
 - discard, 3270
 - max, 3271
 - min, 3271
 - operator<<, 3272
 - operator>>, 3273
 - operator(), 3271
 - operator==, 3272
 - result_type, 3268
 - seed, 3271, 3272
 - shuffle_order_engine, 3269, 3270
- std::slice, 3274
- std::slice_array, 3274
 - operator<=, 3277
 - operator>=, 3277
 - operator*=, 3276
 - operator^=, 3277
 - operator+=, 3276
 - operator-=, 3276
 - operator/=, 3277
 - operator%=, 3276
 - operator&=, 3276
- std::stack, 3277
 - empty, 3280
 - pop, 3280
 - push, 3280
 - size, 3280
 - stack, 3279
 - top, 3280, 3281
- std::student_t_distribution, 3281
 - max, 3283
 - min, 3283
 - operator<<, 3284
 - operator>>, 3285
 - operator(), 3283
 - operator==, 3284
 - param, 3283
 - reset, 3284
 - result_type, 3282
- std::student_t_distribution::param_type, 3285
- std::sub_match, 3287
 - compare, 3288, 3289
 - first, 3291
 - length, 3290

- operator string_type, 3290
- second, 3291
- second_type, 3288
- str, 3290
- std::system_error, 3291
 - what, 3292
- std::this_thread, 811
 - get_id, 811
 - sleep_for, 811
 - sleep_until, 812
 - yield, 812
- std::thread, 3293
 - native_handle, 3294
- std::thread::id, 3294
- std::time_base, 3295
- std::time_get, 3296
 - ~time_get, 3299
 - char_type, 3298
 - date_order, 3299
 - do_date_order, 3300
 - do_get_date, 3300
 - do_get_monthname, 3301
 - do_get_time, 3302
 - do_get_weekday, 3302
 - do_get_year, 3303
 - get_date, 3304
 - get_monthname, 3304
 - get_time, 3305
 - get_weekday, 3306
 - get_year, 3307
 - id, 3308
 - iter_type, 3298
 - time_get, 3299
- std::time_get_byname, 3308
 - char_type, 3310
 - date_order, 3311
 - do_date_order, 3311
 - do_get_date, 3312
 - do_get_monthname, 3312
 - do_get_time, 3313
 - do_get_weekday, 3314
 - do_get_year, 3314
 - get_date, 3315
 - get_monthname, 3316
 - get_time, 3317
 - get_weekday, 3317
 - get_year, 3318
 - id, 3319
 - iter_type, 3310
- std::time_put, 3319
 - ~time_put, 3322
 - char_type, 3321
 - do_put, 3322
 - id, 3324
 - iter_type, 3321
 - put, 3323, 3324
 - time_put, 3322
- std::time_put_byname, 3325
 - char_type, 3326
 - do_put, 3327
 - id, 3329
 - iter_type, 3327
 - put, 3328
- std::timed_mutex, 3329
- std::tr1, 812
- std::tr1::__detail, 816
- std::try_to_lock_t, 3330
- std::tuple, 3330
- std::tuple< _T1 >, 3332
- std::tuple< _T1, _T2 >, 3333
- std::tuple_element< 0, tuple< _Head, _Tail...> >, 3334
- std::tuple_element< __i, tuple< _Head, _Tail...> >, 3334
- std::tuple_size< tuple< _Elements...> >, 3335
- std::type_index, 3335
- std::type_info, 3336
 - ~type_info, 3337
 - name, 3337
- std::unary_function, 3338
 - argument_type, 3339
 - result_type, 3339
- std::unary_negate, 3339
 - argument_type, 3341
 - result_type, 3341
- std::underflow_error, 3342
 - what, 3342
- std::uniform_int_distribution, 3343
 - max, 3344
 - min, 3344
 - operator(), 3345

- param, 3345
- reset, 3346
- result_type, 3344
- uniform_int_distribution, 3344
- std::uniform_int_distribution::param_type, 3346
- std::uniform_real_distribution, 3347
 - max, 3348
 - min, 3348
 - operator(), 3349
 - param, 3349
 - reset, 3349
 - result_type, 3348
 - uniform_real_distribution, 3348
- std::uniform_real_distribution::param_type, 3350
- std::unique_lock, 3351
- std::unique_ptr, 3352
- std::unordered_map, 3355
- std::unordered_multimap, 3357
- std::unordered_multiset, 3360
- std::unordered_set, 3363
- std::uses_allocator, 3366
- std::valarray, 3367
 - valarray, 3370
- std::vector, 3370
 - ~vector, 3377
 - _M_allocate_and_copy, 3378
 - _M_range_check, 3378
 - assign, 3378, 3379
 - at, 3379, 3380
 - back, 3380
 - begin, 3381
 - capacity, 3381
 - cbegin, 3382
 - cend, 3382
 - clear, 3382
 - crbegin, 3382
 - crend, 3382
 - data, 3383
 - emplace, 3383
 - empty, 3383
 - end, 3384
 - erase, 3385
 - front, 3386
 - insert, 3386–3388
 - max_size, 3388
 - operator=, 3389
 - pop_back, 3391
 - push_back, 3391
 - rbegin, 3391, 3392
 - rend, 3392
 - reserve, 3392
 - resize, 3393
 - shrink_to_fit, 3394
 - size, 3394
 - swap, 3394
 - vector, 3374–3377
- std::vector< bool, _Alloc >, 3395
- std::weak_ptr, 3399
- std::weibull_distribution, 3400
 - a, 3401
 - b, 3401
 - max, 3401
 - min, 3402
 - operator(), 3402
 - param, 3402
 - reset, 3403
 - result_type, 3401
- std::weibull_distribution::param_type, 3403
- stdc++.h, 3728
- stddev
 - std::normal_distribution, 3038
- stdexcept, 3728
- stdio_filebuf
 - __gnu_cxx::stdio_filebuf, 1001, 1002
- stdio_filebuf.h, 3729
- stdio_sync_filebuf.h, 3729
- stdtr1c++.h, 3730
- stl_algo.h, 3730
- stl_algobase.h, 3743
- stl_bvector.h, 3746
- stl_construct.h, 3747
- stl_deque.h, 3748
 - _GLIBCXX_DEQUE_BUF_SIZE, 3751
- stl_function.h, 3752
- stl_heap.h, 3754
- stl_iterator.h, 3756
- stl_iterator_base_funcs.h, 3761

- stl_iterator_base_types.h, 3762
- stl_list.h, 3763
- stl_map.h, 3765
- stl_multimap.h, 3766
- stl_multiset.h, 3767
- stl_numeric.h, 3768
- stl_pair.h, 3769
- stl_queue.h, 3770
- stl_raw_storage_iter.h, 3771
- stl_relops.h, 3771
- stl_set.h, 3772
- stl_stack.h, 3773
- stl_tempbuf.h, 3774
- stl_tree.h, 3775
- stl_uninitialized.h, 3776
- stl_vector.h, 3778
- str
 - std::basic_istream, 1993, 1994
 - std::basic_ostringstream, 2142, 2143
 - std::basic_stringbuf, 2281
 - std::basic_stringstream, 2347
 - std::match_results, 2923
 - std::sub_match, 3290
- stream_iterator.h, 3780
- streambuf, 3780
 - io, 65
- streambuf.tcc, 3781
- streambuf_iterator.h, 3782
- streambuf_type
 - std::istreambuf_iterator, 2814
 - std::ostreambuf_iterator, 3134
- streamoff
 - std, 639
- streampos
 - std, 639
- streamsize
 - std, 640
- stride
 - numeric_arrays, 115, 116
- string, 3783
- string_conversions.h, 3786
- string_type
 - std::collate, 2444
 - std::collate_byname, 2451
 - std::messages, 2935
 - std::messages_byname, 2940
 - std::money_get, 2948
 - std::money_put, 2954
 - std::moneypunct, 2961
 - std::moneypunct_byname, 2974
 - std::numpunct, 3113
 - std::numpunct_byname, 3121
- stringbuf
 - io, 65
- stringfwd.h, 3786
- Strings, 292
- strings
 - u32string, 293
- stringstream
 - io, 65
- strstream, 3787
- substr
 - __gnu_cxx::__versa_string, 902
 - __gnu_debug::basic_string, 1159
 - std::basic_string, 2258
- subtractive_rng
 - __gnu_cxx::subtractive_rng, 1062
- suffix
 - std::match_results, 2924
- sum
 - numeric_arrays, 116
- sungetc
 - __gnu_cxx::enc_filebuf, 947
 - __gnu_cxx::stdio_filebuf, 1022
 - __gnu_cxx::stdio_sync_filebuf, 1054
 - std::basic_filebuf, 1639
 - std::basic_streambuf, 2189
 - std::basic_stringbuf, 2282
- swap
 - __gnu_cxx, 399
 - __gnu_cxx::__versa_string, 903
 - __gnu_debug::basic_string, 1160
- futures, 60
- mutating_algorithms, 149
- regex, 266, 267
- std, 732–734
- std::__debug, 747
- std::__profile, 782
- std::basic_regex, 2167
- std::basic_string, 2258
- std::deque, 2586

-
- std::forward_list, 2648
 - std::function<
 - _Res(_- ArgTypes...)>, 2663
 - std::list, 2858
 - std::map, 2908
 - std::match_results, 2924
 - std::multimap, 3003
 - std::multiset, 3023
 - std::set, 3248
 - std::vector, 3394
 - swap_ranges
 - mutating_algorithms, 149
 - sync
 - __gnu_cxx::enc_filebuf, 947
 - __gnu_cxx::stdio_filebuf, 1023
 - __gnu_cxx::stdio_sync_filebuf, 1054
 - std::basic_filebuf, 1639
 - std::basic_fstream, 1708
 - std::basic_ifstream, 1769
 - std::basic_iostream, 1875
 - std::basic_istream, 1933
 - std::basic_istreamstream, 1994
 - std::basic_streambuf, 2190
 - std::basic_stringbuf, 2282
 - std::basic_stringstream, 2347
 - sync_with_stdio
 - std::basic_fstream, 1709
 - std::basic_ifstream, 1770
 - std::basic_ios, 1807
 - std::basic_iostream, 1876
 - std::basic_istream, 1934
 - std::basic_istreamstream, 1995
 - std::basic_ofstream, 2045
 - std::basic_ostream, 2093
 - std::basic_ostreamstream, 2143
 - std::basic_stringstream, 2348
 - std::ios_base, 2750
 - syntax_option_type
 - std::regex_constants, 801
 - system_error, 3787
 - t
 - std::binomial_distribution, 2377
 - table
 - std::ctype< char >, 2495
 - std::ctype_byname< char >, 2544
 - table_size
 - std::ctype< char >, 2498
 - std::ctype_byname< char >, 2547
 - tag_and_trait.hpp, 3788
 - tags.h, 3790
 - tan
 - complex_numbers, 54
 - tanh
 - complex_numbers, 54
 - target
 - std::function<
 - _Res(_- ArgTypes...)>, 2663, 2664
 - target_type
 - std::function<
 - _Res(_- ArgTypes...)>, 2664
 - tellg
 - std::basic_fstream, 1709
 - std::basic_ifstream, 1770
 - std::basic_iostream, 1876
 - std::basic_istream, 1934
 - std::basic_istreamstream, 1995
 - std::basic_stringstream, 2348
 - tellp
 - std::basic_fstream, 1710
 - std::basic_iostream, 1877
 - std::basic_ofstream, 2046
 - std::basic_ostream, 2093
 - std::basic_ostreamstream, 2143
 - std::basic_stringstream, 2349
 - temporary_buffer
 - __gnu_cxx::temporary_buffer, 1064
 - terminate
 - exceptions, 35
 - terminate_handler
 - exceptions, 33
 - test
 - std::bitset, 2392
 - tgmath.h, 3792
 - thousands_sep
 - std::moneypunct, 2971
 - std::moneypunct_byname, 2983
 - std::numpunct, 3118
 - std::numpunct_byname, 3125
 - thread, 3792
 - Threads, 79
-

-
- throw_allocator.h, 3793
 - throw_with_nested
 - exceptions, 36
 - tie
 - std::basic_fstream, 1710, 1711
 - std::basic_ifstream, 1771
 - std::basic_ios, 1807, 1808
 - std::basic_iostream, 1877
 - std::basic_istream, 1935
 - std::basic_istreamstream, 1996
 - std::basic_ofstream, 2046, 2047
 - std::basic_ostream, 2093, 2094
 - std::basic_ostringstream, 2144
 - std::basic_stringstream, 2349, 2350
 - Time, 37
 - time
 - std::locale, 2870
 - time_get
 - std::time_get, 3299
 - time_members.h, 3796
 - time_point_cast
 - std::chrono, 786
 - time_put
 - std::time_put, 3322
 - tinyness_before
 - std::__numeric_limits_base, 1499
 - std::numeric_limits, 3086
 - TLB_size
 - __gnu_parallel::_Settings, 1285
 - to_string
 - std::bitset, 2393
 - to_ulong
 - std::bitset, 2393
 - tolower
 - std, 735
 - std::__ctype_abstract_base, 1403
 - std::ctype, 2480, 2481
 - std::ctype< char >, 2495, 2496
 - std::ctype< wchar_t >, 2513
 - std::ctype_byname, 2530, 2531
 - std::ctype_byname< char >, 2544
 - top
 - std::priority_queue, 3173
 - std::stack, 3280, 3281
 - toupper
 - std, 735
 - std::__ctype_abstract_base, 1404
 - std::ctype, 2481, 2482
 - std::ctype< char >, 2496, 2497
 - std::ctype< wchar_t >, 2514
 - std::ctype_byname, 2531, 2532
 - std::ctype_byname< char >, 2545
 - tr1_math_spec_func
 - assoc_laguerre, 120
 - assoc_legendre, 120
 - beta, 120
 - comp_ellint_1, 120
 - comp_ellint_2, 120
 - comp_ellint_3, 121
 - conf_hyperg, 121
 - cyl_bessel_i, 121
 - cyl_bessel_j, 121
 - cyl_bessel_k, 121
 - cyl_neumann, 122
 - ellint_1, 122
 - ellint_2, 122
 - ellint_3, 122
 - expint, 123
 - hermite, 123
 - hyperg, 123
 - laguerre, 123
 - legendre, 123
 - riemann_zeta, 124
 - sph_bessel, 124
 - sph_legendre, 124
 - sph_neumann, 124
 - traits_type
 - __gnu_cxx::enc_filebuf, 932
 - __gnu_cxx::stdio_filebuf, 1001
 - __gnu_cxx::stdio_sync_filebuf, 1038
 - std::basic_filebuf, 1619
 - std::basic_fstream, 1663, 1664
 - std::basic_ifstream, 1734
 - std::basic_ios, 1792
 - std::basic_iostream, 1832
 - std::basic_istream, 1901
 - std::basic_istream::sentry, 1948
 - std::basic_istreamstream, 1960
 - std::basic_ofstream, 2018
 - std::basic_ostream, 2068
 - std::basic_ostringstream, 2117
-

- std::basic_streambuf, 2173
- std::basic_stringbuf, 2264
- std::basic_stringstream, 2303, 2304
- std::istreambuf_iterator, 2814
- std::ostream_iterator, 3129
- std::ostreambuf_iterator, 3134
- transform
 - mutating_algorithms, 150
 - std::collate, 2448
 - std::collate_byname, 2454
 - std::regex_traits, 3211
- transform_minimal_n
 - __gnu_parallel::_Settings, 1286
- transform_primary
 - std::regex_traits, 3212
- translate
 - std::regex_traits, 3213
- translate_nocase
 - std::regex_traits, 3213
- traps
 - std::__numeric_limits_base, 1499
 - std::numeric_limits, 3086
- tree_policy.hpp, 3796
- tree_trace_base.hpp, 3797
- trie_policy.hpp, 3797
- true_type
 - metaprogramming, 300
- trunename
 - std::numpunct, 3118
 - std::numpunct_byname, 3125
- trunc
 - std::basic_fstream, 1723
 - std::basic_ifstream, 1782
 - std::basic_ios, 1818
 - std::basic_iostream, 1889
 - std::basic_istream, 1946
 - std::basic_istreamstream, 2007
 - std::basic_ofstream, 2057
 - std::basic_ostream, 2105
 - std::basic_ostringstream, 2155
 - std::basic_stringstream, 2362
 - std::ios_base, 2759
- try_lock
 - mutexes, 75
- tuple, 3798
- type_traits, 3800
- type_traits.h, 3806
- type_utils.hpp, 3806
- typeid, 3807
- typeinfo, 3808
- typelist.h, 3808
- types.h, 3810
- types_traits.hpp, 3811
- u16streampos
 - std, 640
- u32streampos
 - std, 640
- u32string
 - strings, 293
- uflow
 - __gnu_cxx::enc_filebuf, 948
 - __gnu_cxx::stdio_filebuf, 1023
 - __gnu_cxx::stdio_sync_filebuf, 1055
 - std::basic_filebuf, 1640
 - std::basic_streambuf, 2190
 - std::basic_stringbuf, 2283
- uncaught_exception
 - exceptions, 36
- underflow
 - __gnu_cxx::enc_filebuf, 948
 - __gnu_cxx::stdio_filebuf, 1024
 - __gnu_cxx::stdio_sync_filebuf, 1055
 - std::basic_filebuf, 1640
 - std::basic_streambuf, 2191
 - std::basic_stringbuf, 2283
- unexpected
 - exceptions, 36
- unexpected_handler
 - exceptions, 33
- unget
 - std::basic_fstream, 1711
 - std::basic_ifstream, 1772
 - std::basic_iostream, 1878
 - std::basic_istream, 1936
 - std::basic_istreamstream, 1997
 - std::basic_stringstream, 2350
- Uniform Distributions, 308
- uniform_int_distribution
 - std::uniform_int_distribution, 3344

- uniform_real_distribution
 - std::uniform_real_distribution, 3348
- uninitialized_copy
 - std, 735
- uninitialized_copy_n
 - SGIextensions, 23
 - std, 736
- uninitialized_fill
 - std, 736
- uninitialized_fill_n
 - std, 737
- unique
 - mutating_algorithms, 151
 - std::forward_list, 2648, 2649
 - std::list, 2859
- unique_copy
 - mutating_algorithms, 152
- unique_copy.h, 3811
- unique_copy_minimal_n
 - __gnu_parallel::_Settings, 1286
- unique_ptr.h, 3811
- unitbuf
 - std, 737
 - std::basic_fstream, 1723
 - std::basic_ifstream, 1782
 - std::basic_ios, 1818
 - std::basic_iostream, 1889
 - std::basic_istream, 1946
 - std::basic_istream, 2007
 - std::basic_ofstream, 2057
 - std::basic_ostream, 2105
 - std::basic_ostringstream, 2155
 - std::basic_stringstream, 2362
 - std::ios_base, 2759
- Unordered Associative, 28
- unordered_map, 3813, 3814
- unordered_map.h, 3816
- unordered_set, 3818, 3819
- unordered_set.h, 3820
- unsetf
 - std::basic_fstream, 1712
 - std::basic_ifstream, 1772
 - std::basic_ios, 1808
 - std::basic_iostream, 1878
 - std::basic_istream, 1936
 - std::basic_istream, 1997
 - std::basic_ofstream, 2047
 - std::basic_ostream, 2094
 - std::basic_ostringstream, 2145
 - std::basic_stringstream, 2351
 - std::ios_base, 2750
- unshift
 - std::__codecvt_abstract_base, 1389
 - std::codecvt, 2417
 - std::codecvt< _InternT, _ExternT, encoding_state >, 2423
 - std::codecvt< char, char, mbstate_t >, 2428
 - std::codecvt< wchar_t, char, mbstate_t >, 2433
 - std::codecvt_byname, 2440
- upper_bound
 - binary_search_algorithms, 206
 - std::map, 2908, 2909
 - std::multimap, 3003, 3004
 - std::multiset, 3023
 - std::set, 3249
- uppercase
 - std, 737
 - std::basic_fstream, 1723
 - std::basic_ifstream, 1782
 - std::basic_ios, 1818
 - std::basic_iostream, 1889
 - std::basic_istream, 1946
 - std::basic_istream, 2007
 - std::basic_ofstream, 2058
 - std::basic_ostream, 2105
 - std::basic_ostringstream, 2155
 - std::basic_stringstream, 2362
 - std::ios_base, 2760
- use_facet
 - std, 737
 - std::locale, 2867
 - std::locale::id, 2874
- Utilities, 80
- utility, 3822
- valarray, 3822
 - numeric_arrays, 93, 94
 - std::valarray, 3370
- valarray_after.h, 3828
- valarray_array.h, 3842

- valarray_array.tcc, 3853
- valarray_before.h, 3854
- value
 - regex, 267
- value_comp
 - std::map, 2909
 - std::multimap, 3004
 - std::multiset, 3024
 - std::set, 3249
- value_compare
 - std::set, 3234
- value_type
 - std::back_insert_iterator, 1606
 - std::complex, 2456
 - std::front_insert_iterator, 2654
 - std::insert_iterator, 2735
 - std::istream_iterator, 2810
 - std::istreambuf_iterator, 2815
 - std::iterator, 2819
 - std::ostream_iterator, 3130
 - std::ostreambuf_iterator, 3134
 - std::raw_storage_iterator, 3192
 - std::reverse_iterator, 3220
 - std::set, 3234
- vector, 3854–3856
 - std::__debug::vector, 1478
 - std::vector, 3374–3377
- vector.tcc, 3857
- vstring.h, 3857
- vstring.tcc, 3862
- vstring_fwd.h, 3863
- vstring_util.h, 3864
- wcerr
 - std, 740
- wcin
 - std, 740
- wclog
 - std, 740
- wcout
 - std, 740
- wcregex_token_iterator
 - regex, 236
- wcsub_match
 - regex, 236
- wfilebuf
 - io, 65
- wfstream
 - io, 65
- what
 - __gnu_cxx::forced_error, 959
 - __gnu_cxx::recursive_init_error, 983
 - std::bad_alloc, 1609
 - std::bad_cast, 1610
 - std::bad_exception, 1611
 - std::bad_function_call, 1612
 - std::bad_typeid, 1613
 - std::bad_weak_ptr, 1614
 - std::domain_error, 2601
 - std::exception, 2609
 - std::future_error, 2674
 - std::invalid_argument, 2739
 - std::ios_base::failure, 2761
 - std::length_error, 2821
 - std::logic_error, 2876
 - std::out_of_range, 3137
 - std::overflow_error, 3139
 - std::range_error, 3182
 - std::regex_error, 3196
 - std::runtime_error, 3226
 - std::system_error, 3292
 - std::underflow_error, 3342
- widen
 - std::__ctype_abstract_base, 1405
 - std::basic_fstream, 1712
 - std::basic_ifstream, 1772
 - std::basic_ios, 1808
 - std::basic_iostream, 1879
 - std::basic_istream, 1936
 - std::basic_istreamstream, 1997
 - std::basic_ofstream, 2047
 - std::basic_ostream, 2095
 - std::basic_ostringstream, 2145
 - std::basic_stringstream, 2351
 - std::ctype, 2482, 2483
 - std::ctype< char >, 2497, 2498
 - std::ctype< wchar_t >, 2515
 - std::ctype_byname, 2532, 2533
 - std::ctype_byname< char >, 2546
- width
 - std::basic_fstream, 1713

- std::basic_ifstream, 1773
- std::basic_ios, 1809
- std::basic_istream, 1879, 1880
- std::basic_istream, 1937
- std::basic_istream, 1998
- std::basic_ofstream, 2048
- std::basic_ostream, 2095, 2096
- std::basic_ostringstream, 2146
- std::basic_stringstream, 2352
- std::ios_base, 2751
- wfstream
 - io, 66
- wios
 - io, 66
- wiostream
 - io, 66
- wistream
 - io, 66
- wstringstream
 - io, 66
- wofstream
 - io, 66
- workstealing.h, 3864
- wostream
 - io, 67
- wostringstream
 - io, 67
- wregex
 - regex, 236
- write
 - std::basic_fstream, 1713
 - std::basic_istream, 1880
 - std::basic_ofstream, 2049
 - std::basic_ostream, 2096
 - std::basic_ostringstream, 2146
 - std::basic_stringstream, 2352
- ws
 - std, 738
- wsregex_token_iterator
 - regex, 237
- wssub_match
 - regex, 237
- wstreambuf
 - io, 67
- wstreampos
 - std, 640
- wstringbuf
 - io, 67
- wstringstream
 - io, 67
- xalloc
 - std::basic_fstream, 1714
 - std::basic_ifstream, 1774
 - std::basic_ios, 1810
 - std::basic_istream, 1881
 - std::basic_istream, 1938
 - std::basic_istream, 1999
 - std::basic_ofstream, 2049
 - std::basic_ostream, 2097
 - std::basic_ostringstream, 2147
 - std::basic_stringstream, 2353
 - std::ios_base, 2751
- xsgn
 - __gnu_cxx::enc_filebuf, 949
 - __gnu_cxx::stdio_filebuf, 1025
 - __gnu_cxx::stdio_sync_filebuf, 1056
 - std::basic_filebuf, 1641
 - std::basic_streambuf, 2192
 - std::basic_stringbuf, 2284
- xspn
 - __gnu_cxx::enc_filebuf, 949
 - __gnu_cxx::stdio_filebuf, 1026
 - __gnu_cxx::stdio_sync_filebuf, 1056
 - std::basic_filebuf, 1642
 - std::basic_streambuf, 2192
 - std::basic_stringbuf, 2285
- yield
 - std::this_thread, 812